UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/074,747	02/11/2002	Pantas Sutardja	MP0096	9964

28285	7590	09/07/2005

KATTEN MUCHIN ROSENMAN LLP (MARVELL)
IP DOCKET
1025 THOMAS JEFFERSON STREET, N.W.
SUITE 700, EAST LOBBY
WASHINGTON, DC 20007-5201

EXAMINER
TORRES, JOSEPH D

ART UNIT	PAPER NUMBER
2133	

DATE MAILED: 09/07/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Application Number: 10/074,747
Filing Date: February 11, 2002
Appellant(s): SUTARDJA ET AL.

Andrew J. Bateman
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 07/18/2005.

(1)   *Real Party in Interest*

A statement identifying the real party in interest is contained in the brief.

(2) *Related Appeals and Interferences*

A statement identifying the related appeals and interferences which will directly

affect or be directly affected by or have a bearing on the decision in the pending appeal

is contained in the brief.

(3) *Status of Claims*

The statement of the status of the claims contained in the brief is correct.

(4) *Status of Amendments After Final*

No amendment after final has been filed.

(5) *Summary of Invention*

The summary of invention contained in the brief is correct.

(6) *Issues*

The appellant's statement of the issues in the brief is correct.

(7) *Grouping of Claims*

The rejection of claims 1-10, 24-33, 38-47, 65-74, 88-97, 102-111, 118-137, 155-

164, and 171-180 stand or fall together because appellant's brief does not include a

statement that this grouping of claims does not stand or fall together and reasons in

support thereof. See 37 CFR 1.192(c)(7).

(8) *Claims Appealed*

The copy of the appealed claims contained in the Appendix to the brief is correct.

(9) *Prior Art of Record*

6,456,208   Nazari et al.   9-2002

(10)   Grounds of Rejection

I. Introduction

The Examiner provides the following introduction to clarify the terminology used in the claim language and to clarify the scope of the claim language.

Claim 1 recites, "obtaining initial binary data having a characteristic Hamming weight; determining the characteristic Hamming weight of the initial binary data; performing a comparison of the characteristic Hamming weight of the initial binary data with a predetermined value; and processing the initial binary data based on the comparison to thereby develop processed binary data having a Hamming weight not less than the characteristic Hamming weight of the initial binary data".

On page 5 of the specification, the Appellant defines Hamming Weight as the number of ones in a binary string. Hence claim 1 substantially recites, --obtaining initial binary data (Note: since any binary data either has zero ones or some other fixed value of ones, any binary data has a characteristic Hamming weight of zero or the fixed number of ones in the binary data if it is non-zero, hence the statement "having a characteristic Hamming weight" is inherent in any binary data); determining the number of ones of the initial binary data; performing a comparison of the number of ones of the initial binary data with a predetermined value; and processing the initial binary data based on the comparison to thereby develop processed binary data having a number of ones not less than the number of ones of the initial binary data--. To provide an example, suppose S = [00001111], then anyone can visually inspect the data to verify that it has a characteristic Hamming weight of 4. It can also be determined that S = [00001111] has

at least 4 ones which is a visual comparison of the Hamming weight to a number 4

(Note: it can also be visually verified that S = [00001111] has more than 3 ones, more

than 2 ones, etc.). If the Examiner processes S = [00001111] by rewriting S =

[00001111] then the rewritten processed S = [00001111] still has a Hamming weight

that is not less than the Hamming weight of the initial unprocessed S = [00001111]

since the Hamming weight of the rewritten processed S = [00001111] has a Hamming

weight of 4 and 4 is not less than 4 since 4 is equal to 4. That is, processing a binary

string S = [00001111] by doing nothing to it is a means for processing the initial binary

data string S = [00001111] based on the observation that the binary data string S =

[00001111] has 4 ones to thereby develop processed binary data string S = [00001111]

having a number of ones not less than the number of ones of the initial binary data.

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 112

The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

Claims 65-74, 88-97 are rejected under 35 U.S.C. 112, first paragraph, as failing to

comply with the enablement requirement. The claim(s) contains subject matter which

was not described in the specification in such a way as to enable one skilled in the art to

which it pertains, or with which it is most nearly connected, to make and/or use the invention. Claim 65 recites, "A computer readable medium having stored thereon". The Examiner asserts that claim 65 substantially recites a computer readable medium for storing the computer program set forth in the body of the claims. There is no indication that the computer program set forth in the body of the claims provides any useful work for the computer readable medium; and nowhere in the specification does the Appellant teach that the body of the claims are directed to providing any useful work for a computer-readable medium.

The Examiner asserts that claim 88 substantially recites a computer readable medium for storing the computer program set forth in the body of the claims. There is no indication that the computer program set forth in the body of the claims provides any useful work for the computer readable medium; and nowhere in the specification does the Appellant teach that the body of the claims are directed to providing any useful work for a computer-readable medium.

The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

Claims 1-10, 24-33, 38-47, 65-74, 88-97, 102-111, 118-137, 155-164 and 171-180 are rejected under 35 U.S.C. 112, second paragraph, as being incomplete for omitting essential elements, such omission amounting to a gap between the elements.

Claim 1 recites, "A communication encoding method". See MPEP § 2172.01. The omitted elements are: how the limitations in the body of the claim are related to communication encoding.

The Appellant contends, "The Patent Office questions 'how the limitations in the body of the claim are related to the communication encoding.' The Patent Office has failed to interpret the claims in light of the disclosure. The Appellants have pointed to specific areas in the specification that support the claims as drafted. The Patent Office asserts that since the preamble of the claims recite a communication encoding apparatus, an 'encoded output' must be explicitly recited in the claims. The Appellants note that each of the aforementioned claims recites the elements or step of 'processing the initial binary data based on the comparison to thereby develop processed binary data having a Hamming weight not less than the characteristic Hamming weight of the initial binary data.' Thus, processed binary data is being developed or otherwise produced according to the steps or features of the claims, and the Patent Office is simply ignoring recited features in the claims".

Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. In re Van Geuns, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993). See MPEP 2145 (VI).

The MPEP requires that the Examiner read the limitations and interpret claim limitations in light of the specification an expressly prohibits reading limitations into the claims.

Claim 1 recites, substantially recites, --obtaining initial binary data (Note: since any binary data either has zero ones or some other fixed value of ones, any binary data has

a characteristic Hamming weight of zero or the fixed number of ones in the binary data if

it is non-zero, hence the statement "having a characteristic Hamming weight" is inherent

in any binary data); determining the number of ones of the initial binary data; performing

a comparison of the number of ones of the initial binary data with a predetermined

value; and processing the initial binary data based on the comparison to thereby

develop processed binary data having a number of ones not less than the number of

ones of the initial binary data--. As an example, suppose S = [00001111], then anyone

can visually inspect the data to verify that it has a characteristic Hamming weight of 4. It

can also be determined that S = [00001111] has at least 4 ones which is a visual

comparison of the Hamming weight to a number 4 (Note: it can also be visually verified

that S = [00001111] has more than 3 ones, more than 2 ones, etc.). If the Examiner

processes S = [00001111] by rewriting S = [00001111] then the rewritten processed S =

[00001111] still has a Hamming weight that is not less than the Hamming weight of the

initial unprocessed S = [00001111] since the Hamming weight of the rewritten

processed S = [00001111] has a Hamming weight of 4 and 4 is not less than 4 since 4

is equal to 4.

The Examiner does not see any connection that the body of claim 1 may have with

communication systems or encoding.

Claim 1, 24, 38, 65, 88, 102, 118, 128, 155 and 171 recite, "processing the initial binary

data based on the comparison". The term "based on" is indefinite since it omits

essential elements necessary to define the relationship between the processing and the comparison.

The Appellant contends, "The Patent Office alleges that: the phrase "based on" is indefinite, because it allegedly 'omits essential elements necessary to define the relationship between the processing and the comparison.' [final Office Action, page 5] It is respectfully submitted that the Patent Office is completely and utterly failing to interpret the claims in light of the disclosure, in derogation of the requirements of 35 U.S.C. § 1 12, second paragraph. Specifically, the Appellants assert that certain sections of the specification support and provide a relationship between 'the processing and comparison,' as recited in the aforementioned claims".

Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. In re Van Geuns, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993). See MPEP 2145 (VI).

The Appellant's remark "interpret the claims in light of the disclosure" refers to language and limitations in the claims, not language and limitations that are not in the claims. Claim 1 recites, "processing the initial binary data based on the comparison to thereby develop processed binary data having a number of ones not less than the number of ones of the initial binary data". The process is not defined. For example, processing a binary string S = [00001111] by doing nothing to it is a means for processing the initial binary data string S = [00001111] based on the observation that the binary data string S = [00001111] has 4 ones to thereby develop processed binary data string S = [00001111] having a number of ones not less than the number of ones of the initial

binary data. The point is the claim language lacks adequate structure to gauge the scope and bounds of the limitation.

The Examiner would also like to point out that the Applicant has not provided any interpretation in the specification to support the Appellant's claim "sections of the specification support and provide a relationship between 'the processing and comparison'". The Appellant is merely pleading, unsupported by proof or a showing of facts.

Claim 38 recites, "A communication encoding apparatus". See MPEP § 2172.01. The omitted elements are: how the limitations in the body of the claim are related to communication encoding.

Claims 10, 33, 47, 74, 97, 111, 127, 137, 164 and 180 recite; "wherein a symbol boundary of an encoded symbol does not change relative to error correction coding." The omitted elements are: how a symbol boundary relates to any of the other data structures such as "initial binary data". The term "does not change relative" is indefinite. The Appellant contends, "The Patent Office alleges that these claims allegedly omit essential elements or steps, particularly "how a symbol boundary relates to any of the other data structures such as `initial binary data. "' [First Office Action, page 5] The Patent Office further asserts that the phrase "does not change relative" is allegedly indefinite. In regard to the former, the Appellants direct the Patent Office to portions of the specification that support the claim language, thus fulfilling the requirements of

M.P.E.P. §2173.02. Further, the Patent Office asserts that the phrase "relative to" is

indefinite".

Although the claims are interpreted in light of the specification, limitations from the

specification are not read into the claims. In re Van Geuns, 988 F.2d 1181, 26

USPQ2d 1057 (Fed. Cir. 1993). See MPEP 2145 (VI).

Claim 10 recites, "a symbol boundary of an encoded symbol does not change relative to

error correction coding". Claim 10 depends from claim 9. Claim 9 does not even

require error correction coding. Even if it did, claim 10 makes no sense. If the encoded

symbol is a result of error correction coding then it is just an encoded symbol whose

boundaries are determined by the error correction coding and if it is not then there is no

relationship between the encoded symbol and the error correction coding. The

language is just plain undecipherable and would always require a translator and unless

the Appellant is going to make himself available to translate the phrase every time

someone should come across the phrase, the Examiner suggest that the Applicant

rewrite the claim.

Claim 65 recites, "A computer readable medium having stored thereon". The Examiner

asserts that claim 65 substantially recites a computer readable medium for storing the

computer program set forth in the body of the claims. There is no indication that the

computer program set forth in the body of the claims provides any useful work for the

computer readable medium; hence the body of the claims are not directed to a

computer-readable medium.

The Examiner asserts that claim 88 substantially recites a computer readable medium

for storing the computer program set forth in the body of the claims. There is no

indication that the computer program set forth in the body of the claims provides any

useful work for the computer readable medium; hence the body of the claims are not

directed to a computer-readable medium.

Claim 102 recites, "A disk drive". See MPEP § 2172.01. The omitted elements are:

how the limitations in the body of the claim are related to a disk drive.

Claim 118 recites, "A communication encoding apparatus". See MPEP § 2172.01. The

omitted elements are: how the limitations in the body of the claim are related to

communication encoding.

Claim 128 recites, "A communication encoding apparatus". See MPEP § 2172.01. The

omitted elements are: how the limitations in the body of the claim are related to

communication encoding.

Claim 155 recites, "A disk drive". See MPEP § 2172.01. The omitted elements are:

how the limitations in the body of the claim are related to a disk drive.

Claim 171 recites, "A communication encoding apparatus". See MPEP § 2172.01. The

omitted elements are: how the limitations in the body of the claim are related to

communication encoding.

Claims 1-10, 24-33, 38-47, 65-74, 88-97, 102-111, 118-137, 155-164 and 171-180 are

rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to

particularly point out and distinctly claim the subject matter which Appellant regards as the invention.

Claim 1, 24, 38, 65, 88, 102, 118, 128, 155 and 171 recite, "processing the initial binary data based on the comparison". The term "based on" is indefinite.

Claim 65 recites, "A computer readable medium having stored thereon". The Examiner asserts that claim 65 substantially recites a computer readable medium for storing the computer program set forth in the body of the claims. There is no indication that the computer program set forth in the body of the claims provides any useful work for the computer readable medium; hence the body of the claims are not directed to a computer-readable medium.

The Examiner asserts that claim 88 substantially recites a computer readable medium for storing the computer program set forth in the body of the claims. There is no indication that the computer program set forth in the body of the claims provides any useful work for the computer readable medium; hence the body of the claims are not directed to a computer-readable medium.

Claim Rejections - 35 USC § 101

35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 1-10, 38-47, 65-74, 88-97, 102-111, 118-137, 155-164 and 171-180 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject

matter. The steps in claim 1 are an abstract algorithm that can be carried out by hand with no link to any tangible process, machine, manufacture, or composition of matter. The steps in claim 38 are an abstract algorithm that can be carried out by hand with no link to any tangible process, machine, manufacture, or composition of matter. Note: The Authoritative Dictionary of IEEE Standards terms defines processor as a system or mechanism that accepts a program as input, prepares it for execution, and executes the processes so defined with data to produce results. Human beings and other computer programs are capable of accepting a program as input, prepares it for execution, and executes the processes so defined with data to produce results. Hence the term "processor" does not cure the problem.

Claim 65 recites, "A computer readable medium having stored thereon". The Examiner asserts that claim 65 substantially recites a computer readable medium for storing the computer program set forth in the body of the claims. Computer programs are non-statutory. There is no indication that the computer program set forth in the body of the claims provides any useful work for the computer readable medium; hence the body of the claims is not directed to a computer-readable medium and the computer-readable medium serves only to store a non-statutory computer program set forth in the body to the claims 65-74.

The Examiner asserts that claim 88 substantially recites a computer readable medium for storing the computer program set forth in the body of the claims. Computer programs are non-statutory. There is no indication that the computer program set forth in the body of the claims provides any useful work for the computer readable medium;

hence the body of the claims is not directed to a computer-readable medium and the computer-readable medium serves only to store a non-statutory computer program set forth in the body to the claims 88-97.

The Applicant contends, "Each of these claims recites, or depends from an independent claims that recites, a computer readable medium "having stored thereon" executable instructions. As such, each claim meets the requirements as discussed in M.P.E.P. §2106 in that '[w]hen functional descriptive material is recorded on some computer-readable medium it becomes structurally and functionally interrelated to the medium and will be statutory in most cases since use of technology permits the function of the descriptive material to be realized.'"

The Examiner disagrees and asserts that M.P.E.P. §2106 also states, "Compare In re Lowry, 32 F.3d 1579, 1583-84, 32 USPQ2d 1031, 1035 (Fed. Cir. 1994) (claim to data structure stored on a computer readable medium that increases computer efficiency held statutory) and Warmerdam, 33 F.3d at 1360-61, 31 USPQ2d at 1759 (**claim to computer having a specific data structure stored in memory held statutory product-by-process claim**) with Warmerdam, 33 F.3d at 1361, 31 USPQ2d at 1760 (claim to a data structure per se held nonstatutory). When nonfunctional descriptive material is recorded on some computer-readable medium, it is not statutory since no requisite functionality is present to satisfy the practical application requirement. Merely claiming nonfunctional descriptive material stored in a computer-readable medium does not make it statutory. Such a result would exalt form over substance. In re Sarkar, 588 F.2d 1330, 1333, 200 USPQ 132, 137 (CCPA 1978) ("[E]ach invention must be

evaluated as claimed; yet semantogenic considerations preclude a determination based solely on words appearing in the claims. In the final analysis under 101, the claimed invention, as a whole, must be evaluated for what it is.") (quoted with approval in Abele, 684 F.2d at 907, 214 USPQ at 687)".

Claim 65 substantially claims a computer program for --obtaining initial binary data (Note: since any binary data either has zero ones or some other fixed value of ones, any binary data has a characteristic Hamming weight of zero or the fixed number of ones in the binary data if it is non-zero, hence the statement "having a characteristic Hamming weight" is inherent in any binary data); determining the number of ones of the initial binary data; performing a comparison of the number of ones of the initial binary data with a predetermined value; and processing the initial binary data based on the comparison to thereby develop processed binary data having a number of ones not less than the number of ones of the initial binary data--.   A computer program is non-statutory. Even without saying one of ordinary skill in the art at the time the invention was made would have known that a computer program can be edited in Wordpefect stored on a floppy. Storing a non-statutory computer program on a floppy does not make it statutory.

The steps in claim 102 are an abstract algorithm that can be carried out by hand with no link to any tangible process, machine, manufacture, or composition of matter.   Note: there is no link to the body of the claims and the disk drive in the preamble of claim 102. In addition, The Authoritative Dictionary of IEEE Standards terms defines processor as a system or mechanism that accepts a program as input, prepares it for execution, and

executes the processes so defined with data to produce results. Human beings and other computer programs are capable of accepting a program as input, prepares it for execution, and executes the processes so defined with data to produce results. Hence the term "processor" does not cure the problem.

The steps in claim 118 are an abstract algorithm that can be carried out by hand with no link to any tangible process, machine, manufacture, or composition of matter.

The steps in claim 128 are an abstract algorithm that can be carried out by hand with no link to any tangible process, machine, manufacture, or composition of matter. Note: there is no link to the body of the claims and the disk drive in the preamble of claim 102. In addition, The Authoritative Dictionary of IEEE Standards terms defines processor as a system or mechanism that accepts a program as input, prepares it for execution, and executes the processes so defined with data to produce results. Human beings and other computer programs are capable of accepting a program as input, prepares it for execution, and executes the processes so defined with data to produce results. Hence the term "processor" does not cure the problem.

The steps in claim 155 are an abstract algorithm that can be carried out by hand with no link to any tangible process, machine, manufacture, or composition of matter. Note: there is no link to the body of the claims and the disk drive in the preamble of claim 102. In addition, The Authoritative Dictionary of IEEE Standards terms defines processor as a system or mechanism that accepts a program as input, prepares it for execution, and executes the processes so defined with data to produce results. Human beings and other computer programs are capable of accepting a program as input, prepares it for

execution, and executes the processes so defined with data to produce results.   Hence

the term "processor" does not cure the problem.

The steps in claim 171 are an abstract algorithm that can be carried out by hand with no

link to any tangible process, machine, manufacture, or composition of matter. Note:

The Authoritative Dictionary of IEEE Standards terms defines processor as a system or

mechanism that accepts a program as input, prepares it for execution, and executes the

processes so defined with data to produce results. Human beings and other computer

programs are capable of accepting a program as input, prepares it for execution, and

executes the processes so defined with data to produce results.   Hence the term

"processor" does not cure the problem.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that

form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

(f) he did not himself invent the subject matter sought to be patented.

Claims 1-10, 24-33, 38-47, 65-74, 88-97, 102-111, 118-137, 155-164, 171-180 are

rejected under 35 U.S.C. 102(e) as being anticipated by Nazari; Nersi et al. (US

6456208 B1, hereafter referred to as Nazari).

The applied reference has a common assignee with the instant application.

Based upon the earlier effective U.S. filing date of the reference, it constitutes prior art

under 35 U.S.C. 102(e). This rejection under 35 U.S.C. 102(e) might be overcome

either by a showing under 37 CFR 1.132 that any invention disclosed but not claimed in

the reference was derived from the inventor of this application and is thus not the

invention "by another," or by an appropriate showing under 37 CFR 1.131.

35 U.S.C. 102(e) rejection of claims 1-3, 24-26, 38-40, 65-67, 88-90, 102-104, 118-120,

155-157, and 171-173.

Nazari teaches a communication encoding method, comprising: obtaining initial binary

data having a characteristic Hamming weight (the Abstract in Nazari teaches that Input

block 200 in Figure 10 obtains initial binary data having a characteristic Hamming

weight greater than 9), determining the characteristic Hamming weight of the initial

binary data (col. 2, lines 5-10 in Nazari teach that the initial 32-bit binary input data is

tested to see if the characteristic Hamming weight of the initial 32-bit input binary data is

greater than 9); performing a comparison of the characteristic Hamming weight of the

initial binary data with a predetermined value (Note: verifying that the initial 32-bit binary

input data has a characteristic Hamming weight greater than 9 is a step for performing a

comparison of the characteristic Hamming weight of the initial 32-bit binary input data

with a predetermined minimum Hamming weight value of 9); and processing the initial

binary data based on the comparison to thereby develop processed binary data having

a Hamming weight not less than the characteristic Hamming weight of the initial binary

data (col. 2, lines 16-21 in Nazari teach that if the initial 32-bit binary input data satisfies the coding constraint of (0, 11/11) and has characteristic Hamming weight greater than 9, then the 16-bit left half of the initial 32-bit binary input data is mapped to the 16-bit left half of the 33-bit binary output data, the 16-bit right half of the initial 32-bit binary input data is mapped to the 16-bit right half of the 33-bit binary output data and the 17th bit of the 33-bit binary output data is set to 1; hence the characteristic Hamming weight of the 33-bit binary output data is one more than that of the input).

Col. 2, lines 22-48 of Nazari teaches that if the input IN does not satisfy the constraint of (0,11/11) and a Hamming weight of at least nine, then IN is broke up into 4 8-bit sequences intLO, intLE, intRO and intRE whereby IN = (intLO, intLE, intRO, intRE) and each of the 8-bit sequences intLO, intLE, intRO and intRE are checked to verify if they match any of the values in Table A of Figure 2 in Nazari and if any of the 8-bit sequences intLO, intLE, intRO and intRE match any of the values in Table A, they are further processed. Any of the 8-bit sequences intLO, intLE, intRO and intRE that do not match the values in Table A are copied to the output OUT as they are. This give rise to 15 additional cases. The Examiner describes the result of each of the cases. The Examiner will use the same notation as in Nazari IN = (intLO, intLE, intRO, intRE) = (in(1), in(2),..., in(32)) and OUT = (out(1), out(2),..., out(34)).

Note in particular; Nazari explicitly teaches that if the coding constraints of (0,11/11) and a Hamming weight of at least nine is satisfied only case 16 below is executed and the rest of the cases are ignored. Col. 2, lines 22-37 in Nazari explicitly teaches that the Tables A and B in Figure 2 are used to analyze the

coding constraints to _determine_ if there is a violation in the coding constraints.

For Example; if IN = 00000010 00000001 00000010 00000001 then the odd and

even interleaves are 0001 0000 0001 0000 and 0000 0001 0000 0001. Neither IN

nor its odd and even interleaves violate the (0,11/11) constraint since there is not

a run of 11 consecutive zeros in either IN or its odd and even interleaves. The

algorithm still flags IN = 00000010 00000001 00000010 00000001 for further

processing since it violates a constraint requirement of at least 9 ones. IN is

converted to OUT= (0, 1, 1, 0, 1, 0, 0, 1, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

out(24), 1, 1, 0, 0, 1, 0, 1, 0, 0, Parity) because it violates a Hamming weight

constraint of 9. The Algorithm of Figures 2-8 explicitly checks for Hamming

weight violations even if there are no (0, 11/11) constraint violations to produce

an output with a Hamming weight of al least 9.

Case 1: _intLO_ is bad, i.e., _intLO_ matches one of the values in Table A.

Col. 2, lines 49-67 teaches that out(18) = 0, out(17)=0, out(16)=1, out(14)=1, out(12)=0,

out(10)=in(17) and Step 33 in Figure 5B teaches that data from Table B corresponding

to the matching value in Table A is copied to (out(2), out(4), out(6), out(8)). The result is

intLE, intRO and intRE are copied as is into OUT = (out(1), out(2),..., out(34)) and the 8-

bit sequence intLO is replaced with the sequence (out(2), out(4), out(6), out(8), out(12),

out(14), out(16), out(18)) = (x1, x2, x3, x4, 0, 1, 1, 0), where x1, x2, x3, x4 are the

replacement values from Table B in Figure 2. One can visually verify that any entry in

Table A will be replaced by a sequence having at least one additional bit. For example

(worst case scenarios): 11000000 will be replaced with (0, 0, 1, 0, 0, 1, 1, 0), 00000011 will be replaced with (0, 1, 0, 0, 0, 1, 1, 0) and 10000001 will be replaced with (1, 0, 0, 0, 0, 1, 1, 0). That is OUT =(out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13), out(14), out(15), out(16), out(17), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26), out(27), out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (in(2), x1, in(4), x2, in(6), x4, in(8), x8, in(10), in(17), in(12), 0, in(14), 1, in(16), 1, 0, 0, in(18), in(19), in(20), in(21), in(22), in(23), in(24), in(25), in(26), in(27), in(28), in(29), in(30), in(31), in(32), Parity).

Result of case 1, OUT which is substantially a reversible representation of IN has a Hamming weight of at least one greater than the Hamming weight of IN.

Case 2: *intLE* is bad, i.e., *intLE* matches one of the values in Table A.

Col. 2, lines 49-67 teaches that out(18) = 0, out(17)=0, out(16)=1, out(14)=0, out(12)=1, out(10)=in(17) and Step 28 in Figure 5A teaches that data from Table B corresponding to the matching value in Table A is copied to (out(2), out(4), out(6), out(8)). The result is intLO, intRO and intRE are copied as is into OUT = (out(1), out(2),..., out(34)) and the 8-bit sequence intLE is replaced with the sequence (out(2), out(4), out(6), out(8), out(12), out(14), out(16), out(18)) = (x1, x2, x3, x4, 1, 0, 1, 0) , where x1, x2, x3, x4 are the replacement values from Table B in Figure 2. One can visually verify that any entry in Table A will be replaced by a sequence having at least one additional bit. For example (worst case scenarios): 11000000 will be replaced with (0, 0, 1, 0, 1, 0, 1, 0), 00000011 will be replaced with (0, 1, 0, 0, 1, 0, 1, 0) and 10000001 will be replaced with

(1, 0, 0, 0, 1, 0, 1, 0). That is OUT =(out(1), out(2), out(3), out(4), out(5), out(6), out(7),

out(8), out(9), out(10), out(11), out(12), out(13), out(14), out(15), out(16), out(17),

out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26), out(27),

out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (in(1), x1, in(3), x2, in(5),

x4, in(7), x8, in(9), in(17), in(11), 1, in(13), 0, in(15), 1, 0, 0, in(18), in(19), in(20), in(21),

in(22), in(23), in(24), in(25), in(26), in(27), in(28), in(29), in(30), in(31), in(32), Parity).

Result of case 2, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least one greater than the Hamming weight of IN.

Case 3: *intRO* is bad, i.e., *intRO* matches one of the values in Table A.

Col. 2, lines 49-67 teaches that out(24) = in(16), out(22)=0, out(20)=1, out(18)=1,

out(17)=0, out(16)=0 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(26), out(28), out(30),

out(32)). The result is intLO, intLE and intRE are copied as is into OUT = (out(1),

out(2),..., out(34)) and the 8-bit sequence intRO is replaced with the sequence (out(16),

out(18), out(20), out(22), out(26), out(28), out(30), out(32)) = (0, 1, 1, 0, x1, x2, x3, x4), ,

where x1, x2, x3, x4 are the replacement values from Table B in Figure 2. One can

visually verify that any entry in Table A will be replaced by a sequence having at least

one additional bit. For example (worst case scenarios): 11000000 will be replaced with

(0, 1, 1, 0, 0, 0, 1, 0,), 00000011 will be replaced with (0, 1, 1, 0, 0, 1, 0, 0,) and

10000001 will be replaced with (0, 1, 1, 0, 1, 0, 0, 0,). That is OUT =(out(1), out(2),

out(3), out(4), out(5), out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13),

out(14), out(15), out(16), out(17), out(18), out(19), out(20), out(21), out(22), out(23),

out(24), out(25), out(26), out(27), out(28), out(29), out(30), out(31), out(32), out(33),

out(34)) = (in(1), in(2), in(3), in(4), in(5), in(6), in(7), in(8), in(9), in(10), in(11), in(12),

in(13), in(14), in(15), 0, 0, 1, in(18), 1, in(20), 0, in(22), in(16), in(24), x1, in(26), x2,

in(28), x3, in(30), x4, in(32), Parity).

Result of case 3, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least one greater than the Hamming weight of IN.

Case 4: *intRE* is bad, i.e., *intRE* matches one of the values in Table A.

Col. 2, lines 49-67 teaches that out(24) = in(16), out(22)=1, out(20)=0, out(18)=1,

out(17)=0, out(16)=0 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(26), out(28), out(30),

out(32)). The result is intLO, intLE and intRO are copied as is into OUT = (out(1),

out(2),..., out(34)) and the 8-bit sequence intRE is replaced with the sequence (out(16),

out(18), out(20), out(22), out(26), out(28), out(30), out(32)) = (0, 1, 0, 1, x1, x2, x3, x4), ,

where x1, x2, x3, x4 are the replacement values from Table B in Figure 2. One can

visually verify that any entry in Table A will be replaced by a sequence having at least

one additional bit. For example (worst case scenarios): 11000000 will be replaced with

(0, 1, 0, 1, 0, 0, 1, 0,), 00000011 will be replaced with (0, 1, 0, 1, 0, 1, 0, 0,) and

10000001 will be replaced with (0, 1, 0, 1, 1, 0, 0, 0,). That is OUT =(out(1), out(2),

out(3), out(4), out(5), out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13),

out(14), out(15), out(16), out(17), out(18), out(19), out(20), out(21), out(22), out(23),

out(24), out(25), out(26), out(27), out(28), out(29), out(30), out(31), out(32), out(33),

out(34)) = (in(1), in(2), in(3), in(4), in(5), in(6), in(7), in(8), in(9), in(10), in(11), in(12),

in(13), in(14), in(15), 0, 0, 1, in(18), 0, in(20), 1, in(22), in(16), in(24), x1, in(26), x2,

in(28), x3, in(30), x4, in(32), Parity).

Result of case 4, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least one greater than the Hamming weight of IN.

Case 5: *intLO* and *intLE* are bad, i.e., *intLO* and *intLE* match one of the values in

Table A.

Col. 2, lines 49-67 teaches that out(18) = 0, out(17)=0, out(16)=1, out(14)=1, out(12)=1,

out(10)=in(17) and Step 33 in Figure 5B teaches that data from Table B corresponding

to the matching value in Table A is copied to (out(2), out(4), out(6), out(8)) and (out(1),

out(3), out(5), out(7)). (1, 1, 1, 1) is copied to (out(9), out(11), out(13), out(15)). The

result is intRO and intRE are copied as is into OUT = (out(1), out(2),..., out(34)) and the

8-bit sequences intLO and intLE are replaced with the sequence (out(1), out(2), out(3),

out(4), out(5), out(6), out(7), out(8), out(9), out(11), out(12), out(13), out(14), out(15),

out(16), out(18)) = (x1, a1, x2, a2, x3, a3, x4, a4, 1, 1, 1, 1, 1, 1, 1, 0), where x1, x2, x3,

x4 and a1, a2, a3, a4 are the replacement values from Table B in Figure 2. One can

visually verify that any entry in Table A will be replaced by a sequence having at least

five additional bits. For example (worst case scenarios): 1100000011000000 will be

replaced with (0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0), 0000001100000011 will be

replaced with (0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0) and 1000000110000001 will be

replaced with (1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0). That is OUT =(out(1), out(2),

out(3), out(4), out(5), out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13),

out(14), out(15), out(16), out(17), out(18), out(19), out(20), out(21), out(22), out(23),

out(24), out(25), out(26), out(27), out(28), out(29), out(30), out(31), out(32), out(33),

out(34)) = (x1, a1, x2, a2, x3, a3, x4, a4, 1, in(17), 1, 1, 1, 1, 1, 1, 0, 0, in(18), in(19),

in(20), in(21), in(22), in(23), in(24), in(25), in(26), in(27), in(28), in(29), in(30), in(31),

in(32), Parity).

Result of case 5, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least five greater than the Hamming weight of IN.

Case 6: *intLO* and *intRO* are bad, i.e., *intLO* and *intRO* match one of the values in

Table A.

Col. 2, lines 49-67 teaches that out(22)=0, out(20)=1, out(18) = 1, out(17)=0, out(16)=1,

out(14)=1, out(12)=0 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(2), out(4), out(6),

out(8)) and (out(26), out(28), out(30), out(32)). The result is intLE and intRE are copied

as is into OUT = (out(1), out(2),..., out(34)) and the 8-bit sequences intLO and intRO

are replaced with the sequence (out(2), out(4), out(6), out(8), out (10), out(12), out(14),

out(16), out(18), out(20), out(22), out(24), out(26), out(28), out(30), out(32)) = (x1, x2,

x3, x4, out (10), 0, 1 ,1 , 1, 1, 0, out(24), a1, a2, a3, a4), where x1, x2, x3, x4 and a1,

a2, a3, a4 are the replacement values from Table B in Figure 2. One can visually verify

that any entry in Table A will be replaced by a sequence having at least five additional

bits. For example (worst case scenarios): 1100000011000000 will be replaced with (0,

0, 1, 0, out (10), 0, 1 ,1 , 1, 1, 0, out(24), 0, 0, 1, 0), 0000001100000011 will be

replaced with (0, 1, 0, 0, out (10), 0, 1 ,1 , 1, 1, 0, out(24), 0, 1, 0, 0) and

1000000110000001 will be replaced with (1, 0, 0, 0, out (10), 0, 1 ,1 , 1, 1, 0, out(24), 1,

0, 0, 0). That is OUT = (out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8),

out(9), out(10), out(11), 1, out(13), 0, out(15), out(16), out(17), out(18), out(19), out(20),

out(21), out(22), out(23), out(24), out(25), out(26), out(27), out(28), out(29), out(30),

out(31), out(32), out(33), out(34)) = (in(2), X1, in(4), X2, in(6), X3, in(8), X4, out(10),

in(17), in(12), 0, in(14), 1, in(16), 1, 1, 1, in(18), 1, in(20), 0, in(22), out(24), in(24), a1,

in(26), a2, in(28), a3, in(30), a4, in(32), Parity). Note: In this case, out(10) and out(24)

can be anything and are not needed to recover the original sequence.

Result of case 6, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least two greater than the Hamming weight of IN.

Case 7: *intLO* and *intRE* are bad, i.e., *intLO* and *intRE* match one of the values in

Table A.

Col. 2, lines 49-67 teaches that out(22)=1, out(20)=0, out(18) = 1, out(17)=0, out(16)=1,

out(14)=1, out(12)=0 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(2), out(4), out(6),

out(8)) and (out(26), out(28), out(30), out(32)). The result is intLE and intRO are copied

as is into OUT = (out(1), out(2),..., out(34)) and the 8-bit sequences intLO and intRE are

replaced with the sequence (out(2), out(4), out(6), out(8), out (10), out(12), out(14),

out(16), out(18), out(20), out(22), out(24), out(26), out(28), out(30), out(32)) = (x1, x2, x3, x4, out (10), 0, 1 ,1 , 1, 0, 1, out(24), a1, a2, a3, a4), where x1, x2, x3, x4 and a1, a2, a3, a4 are the replacement values from Table B in Figure 2. One can visually verify that any entry in Table A will be replaced by a sequence having at least five additional bits. For example (worst case scenarios): 1100000011000000 will be replaced with (0, 0, 1, 0, out (10), 0, 1 ,1 , 1, 0, 1, out(24), 0, 0, 1, 0), 0000001100000011 will be replaced with (0, 1, 0, 0, out (10), 0, 1 ,1 , 1, 0, 1, out(24), 0, 1, 0, 0) and 1000000110000001 will be replaced with (1, 0, 0, 0, out (10), 0, 1 ,1 , 1, 0, 1, out(24), 1, 0, 0, 0). That is OUT = (out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8), out(9), out(10), out(11), 1, out(13), 0, out(15), out(16), out(17), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26), out(27), out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (in(2), X1, in(4), X2, in(6), X3, in(8), X4, out(10), in(17), in(12), 0, in(14), 1, in(16), 1, 1, 1, in(17), 1, in(19), 0, in(21), out(24), in(23), a1, in(25), a2, in(27), a3, in(29), a4, in(31), Parity). Note: In this case, out(10) and out(24) can be anything and are not needed to recover the original sequence.

Result of case 7, OUT which is substantially a reversible representation of IN has a Hamming weight of at least two greater than the Hamming weight of IN.

Case 8: *intLE* and *intRO* are bad, i.e., *intLE* and *intRO* match one of the values in Table A.

Col. 2, lines 49-67 teaches that out(22)=0, out(20)=1, out(18) = 1, out(17)=0, out(16)=1, out(14)=0, out(12)=1 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(2), out(4), out(6),

out(8)) and (out(26), out(28), out(30), out(32)). The result is intLO and intRE are copied

as is into OUT = (out(1), out(2),..., out(34)) and the 8-bit sequences intLE and intRO are

replaced with the sequence (out(2), out(4), out(6), out(8), out (10), out(12), out(14),

out(16), out(18), out(20), out(22), out(24), out(26), out(28), out(30), out(32)) = (x1, x2,

x3, x4, out (10), 1, 0 ,1 , 1, 1, 0, out(24), a1, a2, a3, a4), where x1, x2, x3, x4 and a1,

a2, a3, a4 are the replacement values from Table B in Figure 2. One can visually verify

that any entry in Table A will be replaced by a sequence having at least five additional

bits. For example (worst case scenarios): 1100000011000000 will be replaced with (0,

0, 1, 0, out (10), 1, 0 ,1 , 1, 1, 0, out(24), 0, 0, 1, 0), 0000001100000011 will be

replaced with (0, 1, 0, 0, out (10), 1, 0 ,1 , 1, 1, 0, out(24), 0, 1, 0, 0) and

1000000110000001 will be replaced with (1, 0, 0, 0, out (10), 1, 0 ,1 , 1, 1, 0, out(24), 1,

0, 0, 0). That is OUT = (out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8),

out(9), out(10), out(11), 1, out(13), 0, out(15), out(16), out(17), out(18), out(19), out(20),

out(21), out(22), out(23), out(24), out(25), out(26), out(27), out(28), out(29), out(30),

out(31), out(32), out(33), out(34)) = (in(2), X1, in(4), X2, in(6), X3, in(8), X4, out(10),

in(17), in(12), 1, in(14), 0, in(16), 1, 1, 1, in(18), 1, in(20), 0, in(22), out(24), in(24), a1,

in(26), a2, in(28), a3, in(30), a4, in(32), Parity). Note: In this case, out(10) and out(24)

can be anything and are not needed to recover the original sequence.

Result of case 8, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least two greater than the Hamming weight of IN.

Case 9: **intLE** and **intRE** are bad, i.e., **intLE** and **intRE** match one of the values in

Table A.

Col. 2, lines 49-67 teaches that out(22)=1, out(20)=0, out(18) = 1, out(17)=0, out(16)=1,

out(14)=0, out(12)=1 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(2), out(4), out(6),

out(8)) and (out(26), out(28), out(30), out(32)). The result is intLO and intRO are copied

as is into OUT = (out(1), out(2),..., out(34)) and the 8-bit sequences intLE and intRE are

replaced with the sequence (out(2), out(4), out(6), out(8), out (10), out(12), out(14),

out(16), out(18), out(20), out(22), out(24), out(26), out(28), out(30), out(32)) = (x1, x2,

x3, x4, out (10), 1, 0 ,1 , 1, 0, 1, out(24), a1, a2, a3, a4), where x1, x2, x3, x4 and a1,

a2, a3, a4 are the replacement values from Table B in Figure 2. One can visually verify

that any entry in Table A will be replaced by a sequence having at least five additional

bits. For example (worst case scenarios): 1100000011000000 will be replaced with (0,

0, 1, 0, out (10), 1, 0 ,1 , 1, 0, 1, out(24), 0, 0, 1, 0), 0000001100000011 will be

replaced with (0, 1, 0, 0, out (10), 1, 0 ,1 , 1, 0, 1, out(24), 0, 1, 0, 0) and

1000000110000001 will be replaced with (1, 0, 0, 0, out (10), 1, 0 ,1 , 1, 0, 1, out(24), 1,

0, 0, 0). That is OUT = (out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8),

out(9), out(10), out(11), 1, out(13), 0, out(15), out(16), out(17), out(18), out(19), out(20),

out(21), out(22), out(23), out(24), out(25), out(26), out(27), out(28), out(29), out(30),

out(31), out(32), out(33), out(34)) = (in(2), X1, in(4), X2, in(6), X3, in(8), X4, out(10),

in(17), in(12), 1, in(14), 0, in(16), 1, 1, 1, in(18), 0, in(20), 1, in(22), out(24), in(24), a1,

in(26), a2, in(28), a3, in(30), a4, in(32), Parity). Note: In this case, out(10) and out(24)

can be anything and are not needed to recover the original sequence.

Result of case 9, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least two greater than the Hamming weight of IN.

Case 10: *intRO* and *intRE* are bad, i.e., *intRO* and *intRE* match one of the values in

Table A.

Col. 2, lines 49-67 teaches that out(24) = in(16), out(22)=1, out(20)=1, out(18)=1,

out(17)=0, out(16)=0 and Step 82 in Figure 8 teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(26), out(28), out(30),

out(32)) and (out(27), out(29), out(31), out(33)). (1, 1, 1, 1) is copied to (out(19),

out(21), out(23), out(25)). The result is intLO and intLE are copied as is into OUT =

(out(1), out(2),..., out(34)) and the 8-bit sequences intRO and intRE are replaced with

the sequence (out(16), out(18), out(19), out(20), out(21), out(22), out(23), out(25),

out(26), out(27), out(28), out(29), out(30), out(31), out(32), out(33)) = (0, 1, 1, 1, 1, 1, 1,

1, x1, a1, x2, a2, x3, a3, x4, a4), where x1, x2, x3, x4 and a1, a2, a3, a4 are the

replacement values from Table B in Figure 2. One can visually verify that any entry in

Table A will be replaced by a sequence having at least five additional bits. For example

(worst case scenarios): 1100000011000000 will be replaced with (0, 1, 1, 1, 1, 1, 1, 1,

0, 0, 0, 0, 1, 1, 0, 0), 0000001100000011 will be replaced with (0, 1, 1, 1, 1, 1, 1, 1, 0, 0,

1, 1, 0, 0, 0, 0) and 1000000110000001 will be replaced with (0, 1, 1, 1, 1, 1, 1, 1, 1,

1,0, 0, 0, 0, 0). That is OUT =(out(1), out(2), out(3), out(4), out(5), out(6), out(7),

out(8), out(9), out(10), out(11), out(12), out(13), out(14), out(15), out(16), out(17),

out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26), out(27),

out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (in(1), in(2), in(3), in(4),

in(5), in(6), in(7), in(8), in(9), in(10), in(11), in(12), in(13), in(14), in(15), 0, 0, 1, 1, 1, 1, 1,

1, in(16), out(25), x1, a1, x2, a2, x3, a3, x4, a4, Parity).

Result of case 10, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least five greater than the Hamming weight of IN.

Case 11: *intLO*, *intLE* and *intRO* are bad, i.e., *intLO*, *intLE* and *intRO* match one of

the values in Table A.

Col. 2, lines 49-67 teaches that out(22)=0, out(20)=1, out(18) = 1, out(17)=0, out(16)=1,

out(14)=1, out(12)=1 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(2), out(4), out(6),

out(8)), (out(1), out(3), out(5), out(7)) and (out(26), out(28), out(30), out(32)). (1, 1, 1, 1)

is copied to (out(9), out(11), out(13), out(15)). The result is intRE is copied as is into

OUT = (out(1), out(2),…, out(34)) and the 8-bit sequences intLO, intLE and intRO are

replaced with the sequence (out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8),

out(9), out(10), out(11), out(12), out(13), out(14), out(15), out(16), out(18), out(20),

out(22), out(24), out(26), out(28), out(30), out(32)) = (x1, a1, x2, a2, x3, a3, x4, a4, 1,

out(10), 1, 1, 1, 1, 1, 1, 1, 1, 0, out(24), b1, b2, b3, b4), where (x1, x2, x3, x4), (a1, a2,

a3, a4) and (b1, b2, b3, b4) are the replacement values from Table B in Figure 2. One

can visually verify that any entry in Table A will be replaced by a sequence having at

least six additional bits. For example (worst case scenarios):

110000000110000001100000 will be replaced with (0, 0, 0, 0, 1, 1, 0, 0, 1, out(10), 1, 1,

1, 1, 1, 1, 1, 1, 0, out(24), 0, 0, 1, 0), 000000110000001100000011 will be replaced with

(0, 0, 1, 1, 0, 0, 0, 0, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 0, out(24), 0, 1, 0, 0) and

100000011000000110000001 will be replaced with (1, 1, 0, 0, 0, 0, 0, 0, 1, out(10), 1, 1,

1, 1, 1, 1, 1, 1, 0, out(24), 1, 0, 0, 0). That is OUT =(out(1), out(2), out(3), out(4), out(5),

out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13), out(14), out(15), out(16),

out(17), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26),

out(27), out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (x1, a1, x2, a2,

x3, a3, x4, a4, 1, out(10), 1, 1, 1, 1, 1, 1, 0, 1, in(18), 1, in(20), 0, in(22), out(24), in(24),

b1, in(26), b2, in(28), b3, in(30), b4, in(32), Parity). Note: In this case, out(10) and

out(24) can be anything and are not needed to recover the original sequence.

Result of case 11, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least six greater than the Hamming weight of IN.

Case 12: *intLO*, *intLE* and *intRE* are bad, i.e., *intLO*, *intLE* and *intRE* match one of

the values in Table A.

Col. 2, lines 49-67 teaches that out(22)=0, out(20)=1, out(18) = 1, out(17)=0, out(16)=1,

out(14)=1, out(12)=1 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(2), out(4), out(6),

out(8)), (out(1), out(3), out(5), out(7)) and (out(26), out(28), out(30), out(32)). (1, 1, 1, 1)

is copied to (out(9), out(11), out(13), out(15)). The result is intRO is copied as is into

OUT = (out(1), out(2),..., out(34)) and the 8-bit sequences intLO, intLE and intRE are

replaced with the sequence (out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8),

out(9), out(10), out(11), out(12), out(13), out(14), out(15), out(16), out(18), out(20),

out(22), out(24), out(26), out(28), out(30), out(32)) = (x1, a1, x2, a2, x3, a3, x4, a4, 1,

out(10), 1, 1, 1, 1, 1, 1, 1, 0, 1, out(24), b1, b2, b3, b4), where (x1, x2, x3, x4), (a1, a2,

a3, a4) and (b1, b2, b3, b4) are the replacement values from Table B in Figure 2. One

can visually verify that any entry in Table A will be replaced by a sequence having at

least six additional bits. For example (worst case scenarios):

110000001100000011000000 will be replaced with (0, 0, 0, 0, 1, 1, 0, 0, 1, out(10), 1, 1,

1, 1, 1, 1, 1, 0, 1, out(24), 0, 0, 1, 0), 000000110000001100000011 will be replaced with

(0, 0, 1, 1, 0, 0, 0, 0, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 0, 1, out(24), 0, 1, 0, 0) and

100000011000000110000001 will be replaced with (1, 1, 0, 0, 0, 0, 0, 0, 1, out(10), 1, 1,

1, 1, 1, 1, 1, 0, 1, out(24), 1, 0, 0, 0). That is OUT =(out(1), out(2), out(3), out(4), out(5),

out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13), out(14), out(15), out(16),

out(17), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26),

out(27), out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (x1, a1, x2, a2,

x3, a3, x4, a4, 1, out(10), 1, 1, 1, 1, 1, 1, 0, 1, in(18), 0, in(20), 1, in(22), out(24), in(24),

b1, in(26), b2, in(28), b3, in(30), b4, in(32), Parity). Note: In this case, out(10) and

out(24) can be anything and are not needed to recover the original sequence.

Result of case 12, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least six greater than the Hamming weight of IN.

Case 13: *intLO*, *intRO* and *intRE* are bad, i.e., *intLO*, *intRO* and *intRE* match one of the values in Table A.

Col. 2, lines 49-67 teaches that out(22)=1, out(20)=1, out(18) = 1, out(17)=0, out(16)=1, out(14)=1, out(12)=0 and Step 33 in Figure 5B teaches that data from Table B corresponding to the matching value in Table A is copied to (out(2), out(4), out(6), out(8)), (out(27), out(29), out(31), out(33)) and (out(26), out(28), out(30), out(32)). (1, 1, 1, 1) is copied to (out(19), out(21), out(23), out(25)). The result is intRE is copied as is into OUT = (out(1), out(2),..., out(34)) and the 8-bit sequences intLO, intRO and intRE are replaced with the sequence (out(2), out(4), out(6), out(8), out(10), out(12), out(14), out(16), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26), out(27), out(28), out(29), out(30), out(31), out(32), out(33)) = (x1, x2, x3, x4, out(10), 0, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, a1, b1, a2, b2, a3, b3, a4, b4), where (x1, x2, x3, x4), (a1, a2, a3, a4) and (b1, b2, b3, b4) are the replacement values from Table B in Figure 2. One can visually verify that any entry in Table A will be replaced by a sequence having at least six additional bits. For example (worst case scenarios): 110000001100000011000000 will be replaced with (0, 0, 1, 0, out(10), 0, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 0, 0, 0, 0, 1, 1, 0, 0), 000000110000001100000011 will be replaced with (0, 1, 0, 0, out(10), 0, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 0, 0, 1, 1, 0, 0, 0, 0) and 100000011000000110000001 will be replaced with (1, 0, 0, 0, out(10), 0, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 1, 1, 0, 0, 0, 0, 0, 0). That is OUT =(out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13), out(14), out(15), out(16), out(17), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26),

out(27), out(28), out(29), out(30), out(31), out(32), out(33), out(34))   = (in(2), x1, in(4),

x2, in(6), x3, in(8), x4, in(10), out(10), in(12), 0, in(14), 1, in(16), 1, 0, 1, 1, 1, 1, 1, 1,

out(24), 1, a1, b1, a2, b2, a3, b3, a4, b4, Parity). Note: In this case, out(10) and out(24)

can be anything and are not needed to recover the original sequence.

Result of case 13, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least six greater than the Hamming weight of IN.

Case 14: *intLE*, *intRO* and *intRE* are bad, i.e., *intLE*, *intRO* and *intRE* match one of

the values in Table A.

Col. 2, lines 49-67 teaches that out(22)=1, out(20)=1, out(18) = 1, out(17)=0, out(16)=1,

out(14)=0, out(12)=1 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(2), out(4), out(6),

out(8)), (out(27), out(29), out(31), out(33)) and (out(26), out(28), out(30), out(32)). (1, 1,

1, 1) is copied to (out(19), out(21), out(23), out(25)). The result is intLO is copied as is

into OUT = (out(1), out(2),..., out(34)) and the 8-bit sequences intLE, intRO and intRE

are replaced with the sequence (out(2), out(4), out(6), out(8), out(10), out(12), out(14),

out(16), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26),

out(27), out(28), out(29), out(30), out(31), out(32), out(33)) = (x1, x2, x3, x4, out(10), 1,

0, 1, 1, 1, 1, 1, 1, 1, out(24), 1, a1, b1, a2, b2, a3, b3, a4, b4), where (x1, x2, x3, x4),

(a1, a2, a3, a4) and (b1, b2, b3, b4) are the replacement values from Table B in Figure

2. One can visually verify that any entry in Table A will be replaced by a sequence

having at least six additional bits. For example (worst case scenarios):

1100000001100000011000000 will be replaced with (0, 0, 1, 0, out(10), 1, 0, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 0, 0, 0, 0, 1, 1, 0, 0), 0000001100000001100000011 will be replaced with (0, 1, 0, 0, out(10), 1, 0, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 0, 0, 1, 1, 0, 0, 0, 0) and 1000000110000000110000001 will be replaced with (1, 0, 0, 0, out(10), 1, 0, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 1, 1, 0, 0, 0, 0, 0, 0). That is OUT =(out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13), out(14), out(15), out(16), out(17), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26), out(27), out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (in(2), x1, in(4), x2, in(6), x3, in(8), x4, in(10), out(10), in(12), 1, in(14), 0, in(16), 1, 0, 1, 1, 1, 1, 1, 1, out(24), 1, a1, b1, a2, b2, a3, b3, a4, b4, Parity). Note: In this case, out(10) and out(24) can be anything and are not needed to recover the original sequence.

Result of case 14, OUT which is substantially a reversible representation of IN has a Hamming weight of at least six greater than the Hamming weight of IN.

Case 15: **intLO**, **intLE**, **intRO** and **intRE** are bad, i.e., **intLO**, **intLE**, **intRO** and **intRE** match one of the values in Table A.

Col. 2, lines 49-67 teaches that out(22)=1, out(20)=1, out(18) = 1, out(17)=0, out(16)=1, out(14)=1, out(12)=1 and Step 33 in Figure 5B teaches that data from Table B corresponding to the matching value in Table A is copied to (out(1), out(3), out(5), out(7)),(out(2), out(4), out(6), out(8)), (out(27), out(29), out(31), out(33)) and (out(26), out(28), out(30), out(32)). (1, 1, 1, 1) is copied to (out(9), out(11), out(13), out(15)). (1, 1, 1, 1) is copied to (out(19), out(21), out(23), out(25)). The result is the 8-bit

sequences intLO, intLE, intRO and intRE are replaced with the sequence (out(1), out(2),

out(3), out(4), out(5), out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13),

out(14), out(15), out(16), out(18), out(19), out(20), out(21), out(22), out(23), out(24),

out(25), out(26), out(27), out(28), out(29), out(30), out(31), out(32), out(33)) = (x1, a1,

x2, a2, x3, a3, x4, a4, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, c1, b1, c2,

b2, c3, b3, c4, b4), where (x1, x2, x3, x4), (a1, a2, a3, a4), (b1, b2, b3, b4) and (c1, c2,

c3, c4) are the replacement values from Table B in Figure 2. One can visually verify

that any entry in Table A will be replaced by a sequence having at least six additional

bits. For example (worst case scenarios): 110000001100000011000000011000000 will

be replaced with (0, 0, 0, 0, 1, 1, 0, 0, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, out(24),

1, 0, 0, 0, 0, 1, 1, 0, 0), 00000011000000011000000011000000011 will be replaced with (0,

0, 1, 1, 0, 0, 0, 0, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 0, 0, 1, 1, 0, 0, 0,

0) and 10000001100000011000000110000001 will be replaced with (1, 1, 0, 0, 0, 0, 0,

0, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 1, 1, 0, 0, 0, 0, 0, 0). That is

OUT =(out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8), out(9), out(10), out(11),

out(12), out(13), out(14), out(15), out(16), out(17), out(18), out(19), out(20), out(21),

out(22), out(23), out(24), out(25), out(26), out(27), out(28), out(29), out(30), out(31),

out(32), out(33), out(34)) = (x1, a1, x2, a2, x3, a3, x4, a4, 1, out(10), 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, out(24), 1, a1, b1, a2, b2, a3, b3, a4, b4, Parity). Note: In this case, out(10)

and out(24) can be anything and are not needed to recover the original sequence.

Result of case 15, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least six greater than the Hamming weight of IN.

Case 16: IN satisfies the coding constraints of (0,11/11) and a Hamming weight of at

least nine. OUT =(out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8), out(9),

out(10), out(11), out(12), out(13), out(14), out(15), out(16), out(17), out(18), out(19),

out(20), out(21), out(22), out(23), out(24), out(25), out(26), out(27), out(28), out(29),

out(30), out(31), out(32), out(33), out(34))   =(in(1), in(2), in(3), in(4), in(5), in(6), in(7),

in(8), in(9), in(10), in(11), in(12), in(13), in(14), in(15), in(16), 1, in(17), in(18), in(19),

in(20), in(21), in(22), in(23), in(24), in(25), in(26), in(27), in(28), in(29), in(30), in(31),

in(32), Parity).

Result of case 16, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least 1 greater than the Hamming weight of IN.

Conclusion: The Nazari patent teaches every element of claim 1. Nazari teaches

obtaining initial binary data IN having a characteristic Hamming weight; determining the

characteristic Hamming weight of the initial binary data; performing a comparison of the

characteristic Hamming weight of the initial binary data with a predetermined value (col.

2, lines 5-8 Nazari teaches that the data IN is first tested to verify if its characteristic

hamming weight is greater then or equal to 9); and processing the initial binary data

based on the comparison to thereby develop processed binary data having a Hamming

weight not less than the characteristic Hamming weight of the initial binary data (The

algorithm of Figures 2-8 in Nazari teach that OUT always has a Hamming weight of at

least 1 more than the input IN).

Note in particular; Nazari explicitly teaches that if the coding constraints of (0,11/11) and a Hamming weight of at least nine is satisfied only case 16 below is executed and the rest of the cases are ignored. Col. 2, lines 22-37 in Nazari explicitly teaches that the Tables A and B in Figure 2 are used to analyze the coding constraints to *determine* if there is a violation in the coding constraints. For Example; if IN = 00000010 00000001 00000010 00000001 then the odd and even interleaves are 0001 0000 0001 0000 and 0000 0001 0000 0001. Neither IN nor its odd and even interleaves violate the (0,11/11) constraint since there is not a run of 11 consecutive zeros in either IN or its odd and even interleaves. The algorithm still flags IN = 00000010 00000001 00000010 00000001 for further processing since it violates a constraint requirement of at least 9 ones. IN is converted to OUT= (0, 1, 1, 0, 1, 0, 0, 1, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 1, 0, 0, 1, 0, 1, 0, 0, Parity) because it violates a Hamming weight constraint of 9. The Algorithm of Figures 2-8 explicitly checks for Hamming weight violations even if there are no (0, 11/11) constraint violations to produce an output with a Hamming weight of al least 9.

35 U.S.C. 102(e) rejection of claims 4, 27, 41, 68 91 105 121 131 158 and 174.

The Examiner asserts that when dealing with Binary data replacing a 0 with a 1 or a 1 with a 0 is equivalent to inverting data. For example: if, in Nazari, IN = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), OUT = (1, 1, 1, 1, 1, 1, 1, 1, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 1, 1, 1, 1, 1, 1, 1, 1, Parity).

The zeros in the first nine positions are replaced with ones which is substantially the same as inverting the first nine positions on a bit-by-bit basis since inverting the values in the first nine positions would have the same result, converting the first nine zeros to ones.

35 U.S.C. 102(e) rejection of claims 5, 28, 42, 69, 92, 106 122, 132, 159, and 175. **Note in particular; Nazari explicitly teaches that if the coding constraints of (0,11/11) and a Hamming weight of at least nine is satisfied only case 16 below is executed and the rest of the cases are ignored. Col. 2, lines 22-37 in Nazari explicitly teaches that the Tables A and B in Figure 2 are used to analyze the coding constraints to _determine_ if there is a violation in the coding constraints. For Example; if IN = 00000010 00000001 00000010 00000001 then the odd and even interleaves are 0001 0000 0001 0000 and 0000 0001 0000 0001. Neither IN nor its odd and even interleaves violate the (0,11/11) constraint since there is not a run of 11 consecutive zeros in either IN or its odd and even interleaves. The algorithm still flags IN = 00000010 00000001 00000010 00000001 for further processing since it violates a constraint requirement of at least 9 ones. IN is converted to OUT= (0, 1, 1, 0, 1, 0, 0, 1, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 1, 0, 0, 1, 0, 1, 0, 0, Parity) because it violates a Hamming weight constraint of 9. The Algorithm of Figures 2-8 explicitly checks for Hamming weight violations even if there are no (0, 11/11) constraint violations to produce an output with a Hamming weight of al least 9.**

Results of the analysis are a clear indication of a Hamming constrain violation, that is, if

the number of ones falls below a predetermined value.

35 U.S.C. 102(e) rejection of claims 6, 29, 43, 70, 93, 107, 123, 133, 160, and 176.

OUT(17) is a binary value used to flag violations in the constraints. See Steps 13 and

14 in Figure 4 of Nazari.

35 U.S.C. 102(e) rejection of claims 7, 30, 44, 71, 94, 108, 124, 134, 161, and 177.

The Examiner asserts that when dealing with Binary data replacing a 0 with a 1 or a 1

with a 0 is equivalent to inverting data. For example: if, in Nazari, IN = (0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), OUT = (1, 1, 1, 1, 1,

1, 1, 1, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 1, 1, 1, 1, 1, 1, 1, 1, Parity).

The zeros in the first nine positions are replaced with ones which is substantially the

same as inverting the first nine positions on a bit-by-bit basis since inverting the values

in the first nine positions would have the same result, converting the first nine zeros to

ones.

The Flags out(22), out(20), out(18), out(17), out(16), out(14) and out(12) are indicators

in the output used to indicate which bits have been converted to ensure that the process

is reversible.

35 U.S.C. 102(e) rejection of claims 8, 31, 45, 72, 95, 109, 125, 135, 162, and 178.

The Examiner asserts that when dealing with Binary data replacing a 0 with a 1 or a 1 with a 0 is equivalent to inverting data. For example: if, in Nazari, IN = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), OUT = (1, 1, 1, 1, 1, 1, 1, 1, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 1, 1, 1, 1, 1, 1, 1, 1, Parity). The zeros in the first nine positions are replaced with ones which is substantially the same as inverting the first nine positions on a bit-by-bit basis since inverting the values in the first nine positions would have the same result, converting the first nine zeros to ones.

OUT(17) is a binary value used to flag violations in the constraints. If any of the input bits are converted it will have a vaue of zero.

35 U.S.C. 102(e) rejection of claims 9, 32, 46, 73, 96, 110, 126, 136, 162, and 179.

The Abstract in Nazari teaches Run-Length-Limited (RLL) encoding.

35 U.S.C. 102(e) rejection of claims 10, 33, 47, 74, 97, 111, 127, 137, 163, and 180.

A binary digit is a symbol an the binary digit is both its own end and start point, which never changes regardless of any error correction encoding.

(11)   Response to Argument

I. Introduction

The Examiner provides the following introduction to clarify the terminology used in the claim language and to clarify the scope of the claim language.

Claim 1 recites, "obtaining initial binary data having a characteristic Hamming weight; determining the characteristic Hamming weight of the initial binary data; performing a comparison of the characteristic Hamming weight of the initial binary data with a predetermined value; and processing the initial binary data based on the comparison to thereby develop processed binary data having a Hamming weight not less than the characteristic Hamming weight of the initial binary data".

On page 5 of the specification, the Appellant defines Hamming Weight as the number of ones in a binary string. Hence claim 1 substantially recites, --obtaining initial binary data (Note: since any binary data either has zero ones or some other fixed value of ones, any binary data has a characteristic Hamming weight of zero or the fixed number of ones in the binary data if it is non-zero, hence the statement "having a characteristic Hamming weight" is inherent in any binary data); determining the number of ones of the initial binary data; performing a comparison of the number of ones of the initial binary data with a predetermined value; and processing the initial binary data based on the comparison to thereby develop processed binary data having a number of ones not less than the number of ones of the initial binary data--. To provide an example, suppose S = [00001111], then anyone can visually inspect the data to verify that it has a characteristic Hamming weight of 4. It can also be determined that S = [00001111] has at least 4 ones which is a visual comparison of the Hamming weight to a number 4 (Note: it can also be visually verified that S = [00001111] has more than 3 ones, more than 2 ones, etc.). If the Examiner processes S = [00001111] by rewriting S = [00001111] then the rewritten processed S = [00001111] still has a Hamming weight

that is not less than the Hamming weight of the initial unprocessed S = [00001111]

since the Hamming weight of the rewritten processed S = [00001111] has a Hamming

weight of 4 and 4 is not less than 4 since 4 is equal to 4. That is, processing a binary

string S = [00001111] by doing nothing to it is a means for processing the initial binary

data string S = [00001111] based on the observation that the binary data string S =

[00001111] has 4 ones to thereby develop processed binary data string S = [00001111]

having a number of ones not less than the number of ones of the initial binary data.

II. 35 U.S.C. 112 Rejections

A.1.a (page 12 of the Appellant's Appeal Brief) The Appellant contends, "The Patent

Office questions 'how the limitations in the body of the claim are related to the

communication encoding.' The Patent Office has failed to interpret the claims in light of

the disclosure. The Appellants have pointed to specific areas in the specification that

support the claims as drafted. The Patent Office asserts that since the preamble of the

claims recite a communication encoding apparatus, an 'encoded output' must be

explicitly recited in the claims. The Appellants note that each of the aforementioned

claims recites the elements or step of 'processing the initial binary data based on the

comparison to thereby develop processed binary data having a Hamming weight not

less than the characteristic Hamming weight of the initial binary data.' Thus, processed

binary data is being developed or otherwise produced according to the steps or features

of the claims, and the Patent Office is simply ignoring recited features in the claims".

Although the claims are interpreted in light of the specification, limitations from the

specification are not read into the claims. In re Van Geuns, 988 F.2d 1181, 26

USPQ2d 1057 (Fed. Cir. 1993). See MPEP 2145 (VI).

The MPEP requires that the Examiner read the limitations and interpret claim limitations

in light of the specification an expressly prohibits reading limitations into the claims.

Claim 1 recites, substantially recites, --obtaining initial binary data (Note: since any

binary data either has zero ones or some other fixed value of ones, any binary data has

a characteristic Hamming weight of zero or the fixed number of ones in the binary data if

it is non-zero, hence the statement "having a characteristic Hamming weight" is inherent

in any binary data); determining the number of ones of the initial binary data; performing

a comparison of the number of ones of the initial binary data with a predetermined

value; and processing the initial binary data based on the comparison to thereby

develop processed binary data having a number of ones not less than the number of

ones of the initial binary data--. As an example, suppose S = [00001111], then anyone

can visually inspect the data to verify that it has a characteristic Hamming weight of 4. It

can also be determined that S = [00001111] has at least 4 ones which is a visual

comparison of the Hamming weight to a number 4 (Note: it can also be visually verified

that S = [00001111] has more than 3 ones, more than 2 ones, etc.). If the Examiner

processes S = [00001111] by rewriting S = [00001111] then the rewritten processed S =

[00001111] still has a Hamming weight that is not less than the Hamming weight of the ·

initial unprocessed S = [00001111] since the Hamming weight of the rewritten

processed S = [00001111] has a Hamming weight of 4 and 4 is not less than 4 since 4

is equal to 4. The Examiner still does not see any connection that the body of claim 1

may have with communication systems or encoding.

The Examiner would also like to point out that in the Prior Art, Hamming Weight can be

referred to as a weight, number, count and binary data can be referred to as digital or

pulsed data; hence a valid search for claim 1 should be as follows "((Hamming adj

Weight) weight number count) and ((binary adj data) digital pulsed)". Such a search

produces 781,923 patent documents. The Examiner has not searched all 781,923

patent documents, but has restricted the search to recording medium using RLL codes

and using the term Hamming weight, which is what Examiners do when faced with such

a problem. Since the Examiner found very good art that reads on the Appellant's claim

1, the Examiner has not concluded the search of all 781,923 patent documents.

However should the Appeal board disagree with the Examiner's Prior Art rejection, it is

the Examiners opinion, that the Appellant's independent claims, as written, should not

be allowed until all 781,923 patent documents are searched. The Examiner makes this

point to make the Appellant aware of the search that the Appellant is requiring of the

Examiner by refusing to claim what the Appellant regards as his invention (Note: the

Appellant did not invent counting to verify a count).

Finally with regard to the Applicants contention "Thus, it is respectfully submitted that a

skilled artisan would recognize 'how the limitations in the body of the claim are related

to the communication encoding' when the claims are read in light of the disclosure".

The Question is not whether one of ordinary skill in the art at the time the invention was

made could derive meaning from that Appellant's claims. The question is: what does

the Appellant regard as the Appellant's invention.

A.1.b (page 13 of the Appellant's Appeal Brief). The Appellant contends, "The Patent

Office alleges that: the phrase "based on" is indefinite, because it allegedly 'omits

essential elements necessary to define the relationship between the processing and the

comparison.' [final Office Action, page 5] It is respectfully submitted that the Patent

Office is completely and utterly failing to interpret the claims in light of the disclosure, in

derogation of the requirements of 35 U.S.C. § 1 12, second paragraph. Specifically, the

Appellants assert that certain sections of the specification support and provide a

relationship between 'the processing and comparison,' as recited in the aforementioned

claims".

Although the claims are interpreted in light of the specification, limitations from the

specification are not read into the claims. In re Van Geuns, 988 F.2d 1181, 26

USPQ2d 1057 (Fed. Cir. 1993). See MPEP 2145 (VI).

The Appellant's remark "interpret the claims in light of the disclosure" refers to language

and limitations in the claims, not language and limitations that are not in the claims.

Claim 1 recites, "processing the initial binary data based on the comparison to thereby

develop processed binary data having a number of ones not less than the number of

ones of the initial binary data". The process is not defined. For example, processing a

binary string S = [00001111] by doing nothing to it is a means for processing the initial

binary data string S = [00001111] based on the observation that the binary data string S

= [00001111] has 4 ones to thereby develop processed binary data string S =

[00001111] having a number of ones not less than the number of ones of the initial

binary data. The point is the claim language lacks adequate structure to gauge the

scope and bounds of the limitation.

The Examiner would also like to point out that the Applicant has not provided any

interpretation in the specification to support the Appellant's claim "sections of the

specification support and provide a relationship between 'the processing and

comparison'". The Appellant is merely pleading, unsupported by proof or a showing of

facts.

A.1.c (page 13 of the Appellant's Appeal Brief). The Appellant contends, "The Patent

Office alleges that these claims allegedly omit essential elements or steps, particularly

"how a symbol boundary relates to any of the other data structures such as `initial

binary data. '" [First Office Action, page 5] The Patent Office further asserts that the

phrase "does not change relative" is allegedly indefinite. In regard to the former, the

Appellants direct the Patent Office to portions of the specification that support the claim

language, thus fulfilling the requirements of M.P.E.P. §2173.02. Further, the Patent

Office asserts that the phrase "relative to" is indefinite".

Although the claims are interpreted in light of the specification, limitations from the

specification are not read into the claims. In re Van Geuns, 988 F.2d 1181, 26

USPQ2d 1057 (Fed. Cir. 1993). See MPEP 2145 (VI).

Claim 10 recites, "a symbol boundary of an encoded symbol does not change relative to error correction coding". Claim 10 depends from claim 9. Claim 9 does not even require error correction coding. Even if it did, claim 10 makes no sense. If the encoded symbol is a result of error correction coding then it is just an encoded symbol whose boundaries are determined by the error correction coding and if it is not then there is no relationship between the encoded symbol and the error correction coding. The language is just plain undecipherable and would always require a translator and unless the Appellant is going to make himself available to translate the phrase every time someone should come across the phrase, the Examiner suggest that the Applicant rewrite the claim.

III. 35 U.S.C. 101 Rejections

A.3.a (page 15 of the Appellant's Appeal Brief). The Applicant contends, "The Appellants point to several examples in the M.P.E.P. that support their position that the claims meet the statutory requirements of utility. For example, the Appellants point out that the aforementioned claims accomplish, inter alia, 'the development of processed binary data.'"

Claim 1 recites, substantially recites, --obtaining initial binary data (Note: since any binary data either has zero ones or some other fixed value of ones, any binary data has a characteristic Hamming weight of zero or the fixed number of ones in the binary data if it is non-zero, hence the statement "having a characteristic Hamming weight" is inherent in any binary data); determining the number of ones of the initial binary data; performing

a comparison of the number of ones of the initial binary data with a predetermined

value; and processing the initial binary data based on the comparison to thereby

develop processed binary data having a number of ones not less than the number of

ones of the initial binary data--. As an example, suppose S = [00001111], then anyone

can visually inspect the data to verify that it has a characteristic Hamming weight of 4. It

can also be determined that S = [00001111] has at least 4 ones which is a visual

comparison of the Hamming weight to a number 4 (Note: it can also be visually verified

that S = [00001111] has more than 3 ones, more than 2 ones, etc.). If the Examiner

processes S = [00001111] by rewriting S = [00001111] then the rewritten processed S =

[00001111] still has a Hamming weight that is not less than the Hamming weight of the

initial unprocessed S = [00001111] since the Hamming weight of the rewritten

processed S = [00001111] has a Hamming weight of 4 and 4 is not less than 4 since 4

is equal to 4. S = [00001111] is still processed Binary data produced by handwriting

according to the Applicant's method. The claims are still non-statutory.

A.3.b (page 15 of the Appellant's Appeal Brief). The Applicant contends, "Each of these

claims recites, or depends from an independent claims that recites, a computer

readable medium "having stored thereon" executable instructions. As such, each claim

meets the requirements as discussed in M.P.E.P. §2106 in that '[w]hen functional

descriptive material is recorded on some computer-readable medium it becomes

structurally and functionally interrelated to the medium and will be statutory in most

cases since use of technology permits the function of the descriptive material to be realized.'"

The Examiner disagrees and asserts that M.P.E.P. §2106 also states, "Compare In re Lowry, 32 F.3d 1579, 1583-84, 32 USPQ2d 1031, 1035 (Fed. Cir. 1994) (claim to data structure stored on a computer readable medium that increases computer efficiency held statutory) and Warmerdam, 33 F.3d at 1360-61, 31 USPQ2d at 1759 (**claim to computer having a specific data structure stored in memory held statutory product-by-process claim**) with Warmerdam, 33 F.3d at 1361, 31 USPQ2d at 1760 (claim to a data structure per se held nonstatutory). When nonfunctional descriptive material is recorded on some computer-readable medium, it is not statutory since no requisite functionality is present to satisfy the practical application requirement. Merely claiming nonfunctional descriptive material stored in a computer-readable medium does not make it statutory. Such a result would exalt form over substance. In re Sarkar, 588 F.2d 1330, 1333, 200 USPQ 132, 137 (CCPA 1978) ("[E]ach invention must be evaluated as claimed; yet semantogenic considerations preclude a determination based solely on words appearing in the claims. In the final analysis under 101, the claimed invention, as a whole, must be evaluated for what it is.") (quoted with approval in Abele, 684 F.2d at 907, 214 USPQ at 687)".

Claim 65 substantially claims a computer program for --obtaining initial binary data (Note: since any binary data either has zero ones or some other fixed value of ones, any binary data has a characteristic Hamming weight of zero or the fixed number of ones in the binary data if it is non-zero, hence the statement "having a characteristic

Hamming weight" is inherent in any binary data); determining the number of ones of the

initial binary data; performing a comparison of the number of ones of the initial binary

data with a predetermined value; and processing the initial binary data based on the

comparison to thereby develop processed binary data having a number of ones not less

than the number of ones of the initial binary data--.   A computer program is non-

statutory. Even without saying one of ordinary skill in the art at the time the invention

was made would have known that a computer program can be edited in Wordpefect

stored on a floppy. Storing a non-statutory computer program on a floppy does not

make it statutory.

IV. 35 U.S.C. 102(e) Rejections

A.4.a (page 16 of the Appellant's Appeal Brief). The Applicant contends, "These claims

recite the features of determining the characteristic Hamming weight of the initial binary

data, and performing a comparison of the characteristic Hamming weight of the initial

binary data with a predetermined value. Nazari does not teach or suggest either claim

feature. Nazari teaches the determination of dividing a codeword into four groups, and

each is compared to a look-up table. If the sub-divided group is in the look-up table, a

four bit replacement for that eight but word is selected. In no event does Nazari teach or

suggest the determination of the characteristic Hamming weight of the initial binary

data".

The Appellant is incorrect with respect to the Appellant's characterization or the Nazari

patent. The Appellant goes into great detail about individual steps within the algorithm

of figure 2-8 in Nazari, but fails to recognize the final outcome of the algorithm. In col. 2, lines 1-21 Nazari teaches that a 32 bit input is first checked to verify if "the input word, IN, satisfies the coding constraints of (0,11/11) and a Hamming weight of at least nine" and if it does the "left half of the input word maps directly into the same numbered bit positions in the left half of the output code word, and the right half of input word maps into a numbered bit position one bit higher in order to accommodate out(17) which is set to a logical one to indicate that there was no violations of the coding constraints in the input word". The output OUT has the same bits as the original input IN plus an additional 1 at the out(17) position of the output OUT, hence in this first case OUT has 1 additional one and the Hamming Weight of the output OUT is increased by one. There are fifteen other cases and one of ordinary skill in the art at the time the invention was made could have easily recognized that in each of the fifteen other cases the output OUT is increased by one (the algorithm is just not that difficult for one of ordinary skill in the art at the time the invention was made unless that one of ordinary skill in the art at the time the invention was made purposely chooses to ignore the consequences of the algorithm).

Col. 2, lines 22-48 of Nazari teaches that if the input IN does not satisfy the constraint of (0,11/11) and a Hamming weight of at least nine, then IN is broke up into 4 8-bit sequences intLO, intLE, intRO and intRE whereby IN = (intLO, intLE, intRO, intRE) and each of the 8-bit sequences intLO, intLE, intRO and intRE are checked to verify if they match any of the values in Table A of Figure 2 in Nazari and if any of the 8-bit sequences intLO, intLE, intRO and intRE match any of the values in Table A, they are

further processed. Any of the 8-bit sequences intLO, intLE, intRO and intRE that do not

match the values in Table A are copied to the output OUT as they are. This give rise to

15 additional cases. The Examiner describes the result of each of the cases. The

Examiner will use the same notation as in Nazari IN = (intLO, intLE, intRO, intRE) =

(in(1), in(2),..., in(32)) and OUT = (out(1), out(2),..., out(34)).

Note in particular; Nazari explicitly teaches that if the coding constraints of

(0,11/11) and a Hamming weight of at least nine is satisfied only case 16 below is

executed and the rest of the cases are ignored. Col. 2, lines 22-37 in Nazari

explicitly teaches that the Tables A and B in Figure 2 are used to analyze the

coding constraints to *determine* if there is a violation in the coding constraints.

For Example; if IN = 00000010 00000001 00000010 00000001 then the odd and

even interleaves are 0001 0000 0001 0000 and 0000 0001 0000 0001. Neither IN

nor its odd and even interleaves violate the (0,11/11) constraint since there is not

a run of 11 consecutive zeros in either IN or its odd and even interleaves. The

algorithm still flags IN = 00000010 00000001 00000010 00000001 for further

processing since it violates a constraint requirement of at least 9 ones. IN is

converted to OUT= (0, 1, 1, 0, 1, 0, 0, 1, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

out(24), 1, 1, 0, 0, 1, 0, 1, 0, 0, Parity) because it violates a Hamming weight

constraint of 9. The Algorithm of Figures 2-8 explicitly checks for Hamming

weight violations even if there are no (0, 11/11) constraint violations to produce

an output with a Hamming weight of al least 9.

Case 1: *intLO* is bad, i.e., *intLO* matches one of the values in Table A.

Col. 2, lines 49-67 teaches that out(18) = 0, out(17)=0, out(16)=1, out(14)=1, out(12)=0,

out(10)=in(17) and Step 33 in Figure 5B teaches that data from Table B corresponding

to the matching value in Table A is copied to (out(2), out(4), out(6), out(8)). The result is

intLE, intRO and intRE are copied as is into OUT = (out(1), out(2),..., out(34)) and the 8-

bit sequence intLO is replaced with the sequence (out(2), out(4), out(6), out(8), out(12),

out(14), out(16), out(18)) = (x1, x2, x3, x4, 0, 1, 1, 0), where x1, x2, x3, x4 are the

replacement values from Table B in Figure 2. One can visually verify that any entry in

Table A will be replaced by a sequence having at least one additional bit. For example

(worst case scenarios): intLO=11000000 will be replaced with (0, 0, 1, 0, 0, 1, 1, 0),

intLO=00000011 will be replaced with (0, 1, 0, 0, 0, 1, 1, 0) and intLO=10000001 will be

replaced with (1, 0, 0, 0, 0, 1, 1, 0). That is OUT =(out(1), out(2), out(3), out(4), out(5),

out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13), out(14), out(15), out(16),

out(17), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26),

out(27), out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (in(2), x1, in(4),

x2, in(6), x4, in(8), x8, in(10), in(17), in(12), 0, in(14), 1, in(16), 1, 0, 0, in(18), in(19),

in(20), in(21), in(22), in(23), in(24), in(25), in(26), in(27), in(28), in(29), in(30), in(31),

in(32), Parity).

Result of case 1, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least one greater than the Hamming weight of IN.

Case 2: *intLE* is bad, i.e., *intLE* matches one of the values in Table A.

Col. 2, lines 49-67 teaches that out(18) = 0, out(17)=0, out(16)=1, out(14)=0, out(12)=1,

out(10)=in(17) and Step 28 in Figure 5A teaches that data from Table B corresponding

to the matching value in Table A is copied to (out(2), out(4), out(6), out(8)). The result is

intLO, intRO and intRE are copied as is into OUT = (out(1), out(2),..., out(34)) and the

8-bit sequence intLE is replaced with the sequence (out(2), out(4), out(6), out(8),

out(12), out(14), out(16), out(18)) = (x1, x2, x3, x4, 1, 0, 1, 0) , where x1, x2, x3, x4 are

the replacement values from Table B in Figure 2. One can visually verify that any entry

in Table A will be replaced by a sequence having at least one additional bit. For

example (worst case scenarios): intLE=11000000 will be replaced with (0, 0, 1, 0, 1, 0,

1, 0), intLE=00000011 will be replaced with (0, 1, 0, 0, 1, 0, 1, 0) and intLE=10000001

will be replaced with (1, 0, 0, 0, 1, 0, 1, 0). That is OUT =(out(1), out(2), out(3), out(4),

out(5), out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13), out(14), out(15),

out(16), out(17), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25),

out(26), out(27), out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (in(1), x1,

in(3), x2, in(5), x4, in(7), x8, in(9), in(17), in(11), 1, in(13), 0, in(15), 1, 0, 0, in(18),

in(19), in(20), in(21), in(22), in(23), in(24), in(25), in(26), in(27), in(28), in(29), in(30),

in(31), in(32), Parity).

Result of case 2, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least one greater than the Hamming weight of IN.

Case 3: *intRO* is bad, i.e., *intRO* matches one of the values in Table A.

Col. 2, lines 49-67 teaches that out(24) = in(16), out(22)=0, out(20)=1, out(18)=1,

out(17)=0, out(16)=0 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(26), out(28), out(30),

out(32)). The result is intLO, intLE and intRE are copied as is into OUT = (out(1),

out(2),..., out(34)) and the 8-bit sequence intRO is replaced with the sequence (out(16),

out(18), out(20), out(22), out(26), out(28), out(30), out(32)) = (0, 1, 1, 0, x1, x2, x3, x4), ,

where x1, x2, x3, x4 are the replacement values from Table B in Figure 2. One can

visually verify that any entry in Table A will be replaced by a sequence having at least

one additional bit. For example (worst case scenarios): intRO=11000000 will be

replaced with (0, 1, 1, 0, 0, 0, 1, 0,), intRO=00000011 will be replaced with (0, 1, 1, 0, 0,

1, 0, 0,) and intRO=10000001 will be replaced with (0, 1, 1, 0, 1, 0, 0, 0,). That is OUT

=(out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8), out(9), out(10), out(11),

out(12), out(13), out(14), out(15), out(16), out(17), out(18), out(19), out(20), out(21),

out(22), out(23), out(24), out(25), out(26), out(27), out(28), out(29), out(30), out(31),

out(32), out(33), out(34)) = (in(1), in(2), in(3), in(4), in(5), in(6), in(7), in(8), in(9), in(10),

in(11), in(12), in(13), in(14), in(15), 0, 0, 1, in(18), 1, in(20), 0, in(22), in(16), in(24), x1,

in(26), x2, in(28), x3, in(30), x4, in(32), Parity).

Result of case 3, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least one greater than the Hamming weight of IN.

Case 4: *intRE* is bad, i.e., *intRE* matches one of the values in Table A.

Col. 2, lines 49-67 teaches that out(24) = in(16), out(22)=1, out(20)=0, out(18)=1,

out(17)=0, out(16)=0 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(26), out(28), out(30),

out(32)). The result is intLO, intLE and intRO are copied as is into OUT = (out(1),

out(2),..., out(34)) and the 8-bit sequence intRE is replaced with the sequence (out(16),

out(18), out(20), out(22), out(26), out(28), out(30), out(32)) = (0, 1, 0, 1, x1, x2, x3, x4), ,

where x1, x2, x3, x4 are the replacement values from Table B in Figure 2. One can

visually verify that any entry in Table A will be replaced by a sequence having at least

one additional bit. For example (worst case scenarios): intRE=11000000 will be

replaced with (0, 1, 0, 1, 0, 0, 1, 0,), intRE=00000011 will be replaced with (0, 1, 0, 1, 0,

1, 0, 0,) and intRE=10000001 will be replaced with (0, 1, 0, 1, 1, 0, 0, 0,). That is OUT

=(out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8), out(9), out(10), out(11),

out(12), out(13), out(14), out(15), out(16), out(17), out(18), out(19), out(20), out(21),

out(22), out(23), out(24), out(25), out(26), out(27), out(28), out(29), out(30), out(31),

out(32), out(33), out(34)) = (in(1), in(2), in(3), in(4), in(5), in(6), in(7), in(8), in(9), in(10),

in(11), in(12), in(13), in(14), in(15), 0, 0, 1, in(18), 0, in(20), 1, in(22), in(16), in(24), x1,

in(26), x2, in(28), x3, in(30), x4, in(32), Parity).

Result of case 4, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least one greater than the Hamming weight of IN.

Case 5: *intLO* and *intLE* are bad, i.e., *intLO* and *intLE* match one of the values in

Table A.

Col. 2, lines 49-67 teaches that out(18) = 0, out(17)=0, out(16)=1, out(14)=1, out(12)=1, out(10)=in(17) and Step 33 in Figure 5B teaches that data from Table B corresponding to the matching value in Table A is copied to (out(2), out(4), out(6), out(8)) and (out(1), out(3), out(5), out(7)). (1, 1, 1, 1) is copied to (out(9), out(11), out(13), out(15)). The result is intRO and intRE are copied as is into OUT = (out(1), out(2),..., out(34)) and the 8-bit sequences intLO and intLE are replaced with the sequence (out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8), out(9), out(11), out(12), out(13), out(14), out(15), out(16), out(18)) = (x1, a1, x2, a2, x3, a3, x4, a4, 1, 1, 1, 1, 1, 1, 1, 0), where x1, x2, x3, x4 and a1, a2, a3, a4 are the replacement values from Table B in Figure 2. One can visually verify that any entry in Table A will be replaced by a sequence having at least five additional bits. For example (worst case scenarios): (intLO, intLE)= 1100000011000000 will be replaced with (0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0), (intLO, intLE)= 0000001100000011 will be replaced with (0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0) and (intLO, intLE)= 1000000110000001 will be replaced with (1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0). That is OUT =(out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13), out(14), out(15), out(16), out(17), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26), out(27), out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (x1, a1, x2, a2, x3, a3, x4, a4, 1, in(17), 1, 1, 1, 1, 1, 1, 0, 0, in(18), in(19), in(20), in(21), in(22), in(23), in(24), in(25), in(26), in(27), in(28), in(29), in(30), in(31), in(32), Parity).

Result of case 5, OUT which is substantially a reversible representation of IN has a Hamming weight of at least five greater than the Hamming weight of IN.

Case 6: *intLO* and *intRO* are bad, i.e., *intLO* and *intRO* match one of the values in

Table A.

Col. 2, lines 49-67 teaches that out(22)=0, out(20)=1, out(18) = 1, out(17)=0, out(16)=1,

out(14)=1, out(12)=0 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(2), out(4), out(6),

out(8)) and (out(26), out(28), out(30), out(32)). The result is intLE and intRE are copied

as is into OUT = (out(1), out(2),…, out(34)) and the 8-bit sequences intLO and intRO

are replaced with the sequence (out(2), out(4), out(6), out(8), out (10), out(12), out(14),

out(16), out(18), out(20), out(22), out(24), out(26), out(28), out(30), out(32)) = (x1, x2,

x3, x4, out (10), 0, 1 ,1 , 1, 1, 0, out(24), a1, a2, a3, a4), where x1, x2, x3, x4 and a1,

a2, a3, a4 are the replacement values from Table B in Figure 2. One can visually verify

that any entry in Table A will be replaced by a sequence having at least five additional

bits. For example (worst case scenarios): (intLO, intRO)= 1100000011000000 will be

replaced with (0, 0, 1, 0, out (10), 0, 1 ,1 , 1, 1, 0, out(24), 0, 0, 1, 0), (intLO, intRO)=

0000001100000011 will be replaced with (0, 1, 0, 0, out (10), 0, 1 ,1 , 1, 1, 0, out(24), 0,

1, 0, 0) and (intLO, intRO)= 1000000110000001 will be replaced with (1, 0, 0, 0, out

(10), 0, 1 ,1 , 1, 1, 0, out(24), 1, 0, 0, 0). That is OUT = (out(1), out(2), out(3), out(4),

out(5), out(6), out(7), out(8), out(9), out(10), out(11), 1, out(13), 0, out(15), out(16),

out(17), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26),

out(27), out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (in(2), X1, in(4),

X2, in(6), X3, in(8), X4, out(10), in(17), in(12), 0, in(14), 1, in(16), 1, 1, 1, in(18), 1,

in(20), 0, in(22), out(24), in(24), a1, in(26), a2, in(28), a3, in(30), a4, in(32), Parity).

Note: In this case, out(10) and out(24) can be anything and are not needed to recover

the original sequence.

Result of case 6, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least two greater than the Hamming weight of IN.

Case 7: *intLO* and *intRE* are bad, i.e., *intLO* and *intRE* match one of the values in

Table A.

Col. 2, lines 49-67 teaches that out(22)=1, out(20)=0, out(18) = 1, out(17)=0, out(16)=1,

out(14)=1, out(12)=0 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(2), out(4), out(6),

out(8)) and (out(26), out(28), out(30), out(32)). The result is intLE and intRO are copied

as is into OUT = (out(1), out(2),..., out(34)) and the 8-bit sequences intLO and intRE are

replaced with the sequence (out(2), out(4), out(6), out(8), out (10), out(12), out(14),

out(16), out(18), out(20), out(22), out(24), out(26), out(28), out(30), out(32)) = (x1, x2,

x3, x4, out (10), 0, 1 ,1 , 1, 0, 1, out(24), a1, a2, a3, a4), where x1, x2, x3, x4 and a1,

a2, a3, a4 are the replacement values from Table B in Figure 2. One can visually verify

that any entry in Table A will be replaced by a sequence having at least five additional

bits. For example (worst case scenarios): (intLO, intRE)= 1100000011000000 will be

replaced with (0, 0, 1, 0, out (10), 0, 1 ,1 , 1, 0, 1, out(24), 0, 0, 1, 0), (intLO, intRE)=

0000001100000011 will be replaced with (0, 1, 0, 0, out (10), 0, 1 ,1 , 1, 0, 1, out(24), 0,

1, 0, 0) and (intLO, intRE)= 1000000110000001 will be replaced with (1, 0, 0, 0, out

(10), 0, 1 ,1 , 1, 0, 1, out(24), 1, 0, 0, 0). That is OUT = (out(1), out(2), out(3), out(4),

out(5), out(6), out(7), out(8), out(9), out(10), out(11), 1, out(13), 0, out(15), out(16),

out(17), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26),

out(27), out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (in(2), X1, in(4),

X2, in(6), X3, in(8), X4, out(10), in(17), in(12), 0, in(14), 1, in(16), 1, 1, 1, in(17), 1,

in(19), 0, in(21), out(24), in(23), a1, in(25), a2, in(27), a3, in(29), a4, in(31), Parity).

Note: In this case, out(10) and out(24) can be anything and are not needed to recover

the original sequence.

Result of case 7, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least two greater than the Hamming weight of IN.

Case 8: **intLE** and **intRO** are bad, i.e., **intLE** and **intRO** match one of the values in

Table A.

Col. 2, lines 49-67 teaches that out(22)=0, out(20)=1, out(18) = 1, out(17)=0, out(16)=1,

out(14)=0, out(12)=1 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(2), out(4), out(6),

out(8)) and (out(26), out(28), out(30), out(32)). The result is intLO and intRE are copied

as is into OUT = (out(1), out(2),..., out(34)) and the 8-bit sequences intLE and intRO are

replaced with the sequence (out(2), out(4), out(6), out(8), out (10), out(12), out(14),

out(16), out(18), out(20), out(22), out(24), out(26), out(28), out(30), out(32)) = (x1, x2,

x3, x4, out (10), 1, 0 ,1 , 1, 1, 0, out(24), a1, a2, a3, a4), where x1, x2, x3, x4 and a1,

a2, a3, a4 are the replacement values from Table B in Figure 2. One can visually verify

that any entry in Table A will be replaced by a sequence having at least five additional

bits. For example (worst case scenarios): (intLE, intRO)= 1100000011000000 will be

replaced with (0, 0, 1, 0, out (10), 1, 0 ,1 , 1, 1, 0, out(24), 0, 0, 1, 0), (intLE, intRO)=

0000001100000011 will be replaced with (0, 1, 0, 0, out (10), 1, 0 ,1 , 1, 1, 0, out(24), 0,

1, 0, 0) and (intLE, intRO)= 1000000110000001 will be replaced with (1, 0, 0, 0, out

(10), 1, 0 ,1 , 1, 1, 0, out(24), 1, 0, 0, 0). That is OUT = (out(1), out(2), out(3), out(4),

out(5), out(6), out(7), out(8), out(9), out(10), out(11), 1, out(13), 0, out(15), out(16),

out(17), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26),

out(27), out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (in(2), X1, in(4),

X2, in(6), X3, in(8), X4, out(10), in(17), in(12), 1, in(14), 0, in(16), 1, 1, 1, in(18), 1,

in(20), 0, in(22), out(24), in(24), a1, in(26), a2, in(28), a3, in(30), a4, in(32), Parity).

Note: In this case, out(10) and out(24) can be anything and are not needed to recover

the original sequence.

Result of case 8, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least two greater than the Hamming weight of IN.

Case 9: **intLE** and **intRE** are bad, i.e., **intLE** and **intRE** match one of the values in

Table A.

Col. 2, lines 49-67 teaches that out(22)=1, out(20)=0, out(18) = 1, out(17)=0, out(16)=1,

out(14)=0, out(12)=1 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(2), out(4), out(6),

out(8)) and (out(26), out(28), out(30), out(32)). The result is intLO and intRO are copied

as is into OUT = (out(1), out(2),..., out(34)) and the 8-bit sequences intLE and intRE are

replaced with the sequence (out(2), out(4), out(6), out(8), out (10), out(12), out(14),

out(16), out(18), out(20), out(22), out(24), out(26), out(28), out(30), out(32)) = (x1, x2,

x3, x4, out (10), 1, 0 ,1 , 1, 0, 1, out(24), a1, a2, a3, a4), where x1, x2, x3, x4 and a1,

a2, a3, a4 are the replacement values from Table B in Figure 2. One can visually verify

that any entry in Table A will be replaced by a sequence having at least five additional

bits. For example (worst case scenarios): (intLE, intRE)= 1100000011000000 will be

replaced with (0, 0, 1, 0, out (10), 1, 0 ,1 , 1, 0, 1, out(24), 0, 0, 1, 0), (intLE, intRE)=

0000001100000011 will be replaced with (0, 1, 0, 0, out (10), 1, 0 ,1 , 1, 0, 1, out(24), 0,

1, 0, 0) and (intLE, intRE)= 1000000110000001 will be replaced with (1, 0, 0, 0, out

(10), 1, 0 ,1 , 1, 0, 1, out(24), 1, 0, 0, 0). That is OUT = (out(1), out(2), out(3), out(4),

out(5), out(6), out(7), out(8), out(9), out(10), out(11), 1, out(13), 0, out(15), out(16),

out(17), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26),

out(27), out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (in(2), X1, in(4),

X2, in(6), X3, in(8), X4, out(10), in(17), in(12), 1, in(14), 0, in(16), 1, 1, 1, in(18), 0,

in(20), 1, in(22), out(24), in(24), a1, in(26), a2, in(28), a3, in(30), a4, in(32), Parity).

Note: In this case, out(10) and out(24) can be anything and are not needed to recover

the original sequence.

Result of case 9, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least two greater than the Hamming weight of IN.

Case 10: *intRO* and *intRE* are bad, i.e., *intRO* and *intRE* match one of the values in

Table A.

Col. 2, lines 49-67 teaches that out(24) = in(16), out(22)=1, out(20)=1, out(18)=1,

out(17)=0, out(16)=0 and Step 82 in Figure 8 teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(26), out(28), out(30),

out(32)) and (out(27), out(29), out(31), out(33)). (1, 1, 1, 1) is copied to (out(19),

out(21), out(23), out(25)). The result is intLO and intLE are copied as is into OUT =

(out(1), out(2),..., out(34)) and the 8-bit sequences intRO and intRE are replaced with

the sequence (out(16), out(18), out(19), out(20), out(21), out(22), out(23), out(25),

out(26), out(27), out(28), out(29), out(30), out(31), out(32), out(33)) = (0, 1, 1, 1, 1, 1, 1,

1, x1, a1, x2, a2, x3, a3, x4, a4), where x1, x2, x3, x4 and a1, a2, a3, a4 are the

replacement values from Table B in Figure 2. One can visually verify that any entry in

Table A will be replaced by a sequence having at least five additional bits. For example

(worst case scenarios): (intRO, intRE)= 1100000011000000 will be replaced with (0, 1,

1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0), (intRO, intRE)= 0000001100000011 will be

replaced with (0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0) and (intRO, intRE)=

1000000110000001 will be replaced with (0, 1, 1, 1, 1, 1, 1, 1, 1, 1,0, 0, 0, 0, 0, 0). That

is OUT =(out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8), out(9), out(10),

out(11), out(12), out(13), out(14), out(15), out(16), out(17), out(18), out(19), out(20),

out(21), out(22), out(23), out(24), out(25), out(26), out(27), out(28), out(29), out(30),

out(31), out(32), out(33), out(34)) = (in(1), in(2), in(3), in(4), in(5), in(6), in(7), in(8),

in(9), in(10), in(11), in(12), in(13), in(14), in(15), 0, 0, 1, 1, 1, 1, 1, 1, in(16), out(25), x1, a1, x2, a2, x3, a3, x4, a4, Parity).

Result of case 10, OUT which is substantially a reversible representation of IN has a Hamming weight of at least five greater than the Hamming weight of IN.

Case 11: *intLO*, *intLE* and *intRO* are bad, i.e., *intLO*, *intLE* and *intRO* match one of the values in Table A.

Col. 2, lines 49-67 teaches that out(22)=0, out(20)=1, out(18) = 1, out(17)=0, out(16)=1, out(14)=1, out(12)=1 and Step 33 in Figure 5B teaches that data from Table B corresponding to the matching value in Table A is copied to (out(2), out(4), out(6), out(8)), (out(1), out(3), out(5), out(7)) and (out(26), out(28), out(30), out(32)). (1, 1, 1, 1) is copied to (out(9), out(11), out(13), out(15)). The result is intRE is copied as is into OUT = (out(1), out(2),..., out(34)) and the 8-bit sequences intLO, intLE and intRO are replaced with the sequence (out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13), out(14), out(15), out(16), out(18), out(20), out(22), out(24), out(26), out(28), out(30), out(32)) = (x1, a1, x2, a2, x3, a3, x4, a4, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 0, out(24), b1, b2, b3, b4), where (x1, x2, x3, x4), (a1, a2, a3, a4) and (b1, b2, b3, b4) are the replacement values from Table B in Figure 2. One can visually verify that any entry in Table A will be replaced by a sequence having at least six additional bits. For example (worst case scenarios): (intLO, intLE, intRO)= 110000001100000011000000 will be replaced with (0, 0, 0, 0, 1, 1, 0, 0, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 0, out(24), 0, 0, 1, 0), (intLO, intLE, intRO)=

00000011000000110000011 will be replaced with (0, 0, 1, 1, 0, 0, 0, 0, 1, out(10), 1, 1,

1, 1, 1, 1, 1, 1, 0, out(24), 0, 1, 0, 0) and (intLO, intLE, intRO)=

100000011000000110000001 will be replaced with (1, 1, 0, 0, 0, 0, 0, 0, 1, out(10), 1, 1,

1, 1, 1, 1, 1, 1, 0, out(24), 1, 0, 0, 0). That is OUT =(out(1), out(2), out(3), out(4), out(5),

out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13), out(14), out(15), out(16),

out(17), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26),

out(27), out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (x1, a1, x2, a2,

x3, a3, x4, a4, 1, out(10), 1, 1, 1, 1, 1, 1, 0, 1, in(18), 1, in(20), 0, in(22), out(24), in(24),

b1, in(26), b2, in(28), b3, in(30), b4, in(32), Parity). Note: In this case, out(10) and

out(24) can be anything and are not needed to recover the original sequence.

Result of case 11, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least six greater than the Hamming weight of IN.

Case 12: *intLO*, *intLE* and *intRE* are bad, i.e., *intLO*, *intLE* and *intRE* match one of

the values in Table A.

Col. 2, lines 49-67 teaches that out(22)=0, out(20)=1, out(18) = 1, out(17)=0, out(16)=1,

out(14)=1, out(12)=1 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(2), out(4), out(6),

out(8)), (out(1), out(3), out(5), out(7)) and (out(26), out(28), out(30), out(32)). (1, 1, 1, 1)

is copied to (out(9), out(11), out(13), out(15)). The result is intRO is copied as is into

OUT = (out(1), out(2),…, out(34)) and the 8-bit sequences intLO, intLE and intRE are

replaced with the sequence (out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8),

out(9), out(10), out(11), out(12), out(13), out(14), out(15), out(16), out(18), out(20),

out(22), out(24), out(26), out(28), out(30), out(32)) = (x1, a1, x2, a2, x3, a3, x4, a4, 1,

out(10), 1, 1, 1, 1, 1, 1, 1, 0, 1, out(24), b1, b2, b3, b4), where (x1, x2, x3, x4), (a1, a2,

a3, a4) and (b1, b2, b3, b4) are the replacement values from Table B in Figure 2. One

can visually verify that any entry in Table A will be replaced by a sequence having at

least six additional bits. For example (worst case scenarios): (intLO, intLE, intRE)=

110000000110000000011000000 will be replaced with (0, 0, 0, 0, 1, 1, 0, 0, 1, out(10), 1, 1,

1, 1, 1, 1, 1, 0, 1, out(24), 0, 0, 1, 0), (intLO, intLE, intRE)=

000000011000000011000000011 will be replaced with (0, 0, 1, 1, 0, 0, 0, 0, 1, out(10), 1, 1,

1, 1, 1, 1, 1, 0, 1, out(24), 0, 1, 0, 0) and (intLO, intLE, intRE)=

100000001100000001100000001 will be replaced with (1, 1, 0, 0, 0, 0, 0, 0, 1, out(10), 1, 1,

1, 1, 1, 1, 1, 0, 1, out(24), 1, 0, 0, 0). That is OUT =(out(1), out(2), out(3), out(4), out(5),

out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13), out(14), out(15), out(16),

out(17), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26),

out(27), out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (x1, a1, x2, a2,

x3, a3, x4, a4, 1, out(10), 1, 1, 1, 1, 1, 1, 0, 1, in(18), 0, in(20), 1, in(22), out(24), in(24),

b1, in(26), b2, in(28), b3, in(30), b4, in(32), Parity). Note: In this case, out(10) and

out(24) can be anything and are not needed to recover the original sequence.

Result of case 12, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least six greater than the Hamming weight of IN.

Case 13: *intLO*, *intRO* and *intRE* are bad, i.e., *intLO*, *intRO* and *intRE* match one of

the values in Table A.

Col. 2, lines 49-67 teaches that out(22)=1, out(20)=1, out(18) = 1, out(17)=0, out(16)=1,

out(14)=1, out(12)=0 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(2), out(4), out(6),

out(8)), (out(27), out(29), out(31), out(33)) and (out(26), out(28), out(30), out(32)). (1, 1,

1, 1) is copied to (out(19), out(21), out(23), out(25)). The result is intRE is copied as is

into OUT = (out(1), out(2),..., out(34)) and the 8-bit sequences intLO, intRO and intRE

are replaced with the sequence (out(2), out(4), out(6), out(8), out(10), out(12), out(14),

out(16), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26),

out(27), out(28), out(29), out(30), out(31), out(32), out(33)) = (x1, x2, x3, x4, out(10), 0,

1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, a1, b1, a2, b2, a3, b3, a4, b4), where (x1, x2, x3, x4),

(a1, a2, a3, a4) and (b1, b2, b3, b4) are the replacement values from Table B in Figure

2. One can visually verify that any entry in Table A will be replaced by a sequence

having at least six additional bits. For example (worst case scenarios): (intLO, intRO,

intRE)= 110000001100000011000000 will be replaced with (0, 0, 1, 0, out(10), 0, 1, 1,

1, 1, 1, 1, 1, 1, out(24), 1, 0, 0, 0, 0, 1, 1, 0, 0), (intLO, intRO, intRE)=

000000110000001100000011 will be replaced with (0, 1, 0, 0, out(10), 0, 1, 1, 1, 1, 1, 1,

1, 1, out(24), 1, 0, 0, 1, 1, 0, 0, 0, 0) and (intLO, intRO, intRE)=

100000011000000110000001 will be replaced with (1, 0, 0, 0, out(10), 0, 1, 1, 1, 1, 1, 1,

1, 1, out(24), 1, 1, 1, 0, 0, 0, 0, 0, 0). That is OUT =(out(1), out(2), out(3), out(4), out(5),

out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13), out(14), out(15), out(16),

out(17), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26),

out(27), out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (in(2), x1, in(4),

x2, in(6), x3, in(8), x4, in(10), out(10), in(12), 0, in(14), 1, in(16), 1, 0, 1, 1, 1, 1, 1, 1,

out(24), 1, a1, b1, a2, b2, a3, b3, a4, b4, Parity). Note: In this case, out(10) and out(24)

can be anything and are not needed to recover the original sequence.

Result of case 13, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least six greater than the Hamming weight of IN.

Case 14: *intLE*, *intRO* and *intRE* are bad, i.e., *intLE*, *intRO* and *intRE* match one of

the values in Table A.

Col. 2, lines 49-67 teaches that out(22)=1, out(20)=1, out(18) = 1, out(17)=0, out(16)=1,

out(14)=0, out(12)=1 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(2), out(4), out(6),

out(8)), (out(27), out(29), out(31), out(33)) and (out(26), out(28), out(30), out(32)). (1, 1,

1, 1) is copied to (out(19), out(21), out(23), out(25)). The result is intLO is copied as is

into OUT = (out(1), out(2),…, out(34)) and the 8-bit sequences intLE, intRO and intRE

are replaced with the sequence (out(2), out(4), out(6), out(8), out(10), out(12), out(14),

out(16), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26),

out(27), out(28), out(29), out(30), out(31), out(32), out(33)) = (x1, x2, x3, x4, out(10), 1,

0, 1, 1, 1, 1, 1, 1, 1, out(24), 1, a1, b1, a2, b2, a3, b3, a4, b4), where (x1, x2, x3, x4),

(a1, a2, a3, a4) and (b1, b2, b3, b4) are the replacement values from Table B in Figure

2. One can visually verify that any entry in Table A will be replaced by a sequence

having at least six additional bits. For example (worst case scenarios): (intLE, intRO,

intRE)= 110000001100000011000000 will be replaced with (0, 0, 1, 0, out(10), 1, 0, 1,

1, 1, 1, 1, 1, 1, out(24), 1, 0, 0, 0, 0, 1, 1, 0, 0), (intLE, intRO, intRE)=

000000110000001100000011 will be replaced with (0, 1, 0, 0, out(10), 1, 0, 1, 1, 1, 1, 1,

1, 1, out(24), 1, 0, 0, 1, 1, 0, 0, 0, 0) and (intLE, intRO, intRE)=

100000011000000110000001 will be replaced with (1, 0, 0, 0, out(10), 1, 0, 1, 1, 1, 1, 1,

1, 1, out(24), 1, 1, 1, 0, 0, 0, 0, 0, 0). That is OUT =(out(1), out(2), out(3), out(4), out(5),

out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13), out(14), out(15), out(16),

out(17), out(18), out(19), out(20), out(21), out(22), out(23), out(24), out(25), out(26),

out(27), out(28), out(29), out(30), out(31), out(32), out(33), out(34)) = (in(2), x1, in(4),

x2, in(6), x3, in(8), x4, in(10), out(10), in(12), 1, in(14), 0, in(16), 1, 0, 1, 1, 1, 1, 1, 1,

out(24), 1, a1, b1, a2, b2, a3, b3, a4, b4, Parity). Note: In this case, out(10) and out(24)

can be anything and are not needed to recover the original sequence.

Result of case 14, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least six greater than the Hamming weight of IN.

Case 15: *intLO*, *intLE*, *intRO* and *intRE* are bad, i.e., *intLO*, *intLE*, *intRO* and *intRE*

match one of the values in Table A.

Col. 2, lines 49-67 teaches that out(22)=1, out(20)=1, out(18) = 1, out(17)=0, out(16)=1,

out(14)=1, out(12)=1 and Step 33 in Figure 5B teaches that data from Table B

corresponding to the matching value in Table A is copied to (out(1), out(3), out(5),

out(7)),(out(2), out(4), out(6), out(8)), (out(27), out(29), out(31), out(33)) and (out(26),

out(28), out(30), out(32)). (1, 1, 1, 1) is copied to (out(9), out(11), out(13), out(15)). (1,

1, 1, 1) is copied to (out(19), out(21), out(23), out(25)). The result is the 8-bit

sequences intLO, intLE, intRO and intRE are replaced with the sequence (out(1), out(2),

out(3), out(4), out(5), out(6), out(7), out(8), out(9), out(10), out(11), out(12), out(13),

out(14), out(15), out(16), out(18), out(19), out(20), out(21), out(22), out(23), out(24),

out(25), out(26), out(27), out(28), out(29), out(30), out(31), out(32), out(33)) = (x1, a1,

x2, a2, x3, a3, x4, a4, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, c1, b1, c2,

b2, c3, b3, c4, b4), where (x1, x2, x3, x4), (a1, a2, a3, a4), (b1, b2, b3, b4) and (c1, c2,

c3, c4) are the replacement values from Table B in Figure 2. One can visually verify

that any entry in Table A will be replaced by a sequence having at least six additional

bits. For example (worst case scenarios): (intLO, intLE, intRO, intRE)=

11000000110000001100000011000000 will be replaced with (0, 0, 0, 0, 1, 1, 0, 0, 1,

out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 0, 0, 0, 0, 1, 1, 0, 0),

00000011000000110000001100000011 will be replaced with (0, 0, 1, 1, 0, 0, 0, 0, 1,

out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 0, 0, 1, 1, 0, 0, 0, 0) and

10000001100000011000000110000001 will be replaced with (1, 1, 0, 0, 0, 0, 0, 0, 1,

out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 1, 1, 0, 0, 0, 0, 0, 0). That is OUT

=(out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8), out(9), out(10), out(11),

out(12), out(13), out(14), out(15), out(16), out(17), out(18), out(19), out(20), out(21),

out(22), out(23), out(24), out(25), out(26), out(27), out(28), out(29), out(30), out(31),

out(32), out(33), out(34)) = (x1, a1, x2, a2, x3, a3, x4, .a4, 1, out(10), 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, out(24), 1, a1, b1, a2, b2, a3, b3, a4, b4, Parity). Note: In this case, out(10)

and out(24) can be anything and are not needed to recover the original sequence.

Result of case 15, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least six greater than the Hamming weight of IN.

Case 16: IN satisfies the coding constraints of (0,11/11) and a Hamming weight of at

least nine. OUT =(out(1), out(2), out(3), out(4), out(5), out(6), out(7), out(8), out(9),

out(10), out(11), out(12), out(13), out(14), out(15), out(16), out(17), out(18), out(19),

out(20), out(21), out(22), out(23), out(24), out(25), out(26), out(27), out(28), out(29),

out(30), out(31), out(32), out(33), out(34))   =(in(1), in(2), in(3), in(4), in(5), in(6), in(7),

in(8), in(9), in(10), in(11), in(12), in(13), in(14), in(15), in(16), 1, in(17), in(18), in(19),

in(20), in(21), in(22), in(23), in(24), in(25), in(26), in(27), in(28), in(29), in(30), in(31),

in(32), Parity).

Result of case 16, OUT which is substantially a reversible representation of IN has a

Hamming weight of at least 1 greater than the Hamming weight of IN.

Conclusion: The Nazari patent teaches every element of claim 1. Nazari teaches

obtaining initial binary data IN having a characteristic Hamming weight; determining the

characteristic Hamming weight of the initial binary data; performing a comparison of the

characteristic Hamming weight of the initial binary data with a predetermined value (col.

2, lines 5-8 Nazari teaches that the data IN is first tested to verify if the coding

constraints of (0,11/11) and a Hamming weight of at least nine is satisfied); and

processing the initial binary data based on the comparison to thereby develop

processed binary data having a Hamming weight not less than the characteristic

Hamming weight of the initial binary data (The algorithm of Figures 2-8 in Nazari teach

that OUT always has a Hamming weight of at least 1 more than the input IN).

Note in particular; Nazari explicitly teaches that if the coding constraints of (0,11/11) and

a <u>Hamming weight</u> of at least nine is satisfied only case 16 above is executed and the

rest of the cases are ignored.

<u>Note in particular; Nazari explicitly teaches that if the coding constraints of</u>

<u>(0,11/11) and a Hamming weight of at least nine is satisfied only case 16 below is</u>

<u>executed and the rest of the cases are ignored. Col. 2, lines 22-37 in Nazari</u>

<u>explicitly teaches that the Tables A and B in Figure 2 are used to analyze the</u>

<u>coding constraints to *determine* if there is a violation in the coding constraints.</u>

<u>For Example; if IN = 00000010 00000001 00000010 00000001 then the odd and</u>

<u>even interleaves are 0001 0000 0001 0000 and 0000 0001 0000 0001. Neither IN</u>

<u>nor its odd and even interleaves violate the (0,11/11) constraint since there is not</u>

<u>a run of 11 consecutive zeros in either IN or its odd and even interleaves. The</u>

<u>algorithm still flags IN = 00000010 00000001 00000010 00000001 for further</u>

<u>processing since it violates a constraint requirement of at least 9 ones. IN is</u>

<u>converted to OUT= (0, 1, 1, 0, 1, 0, 0, 1, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,</u>

<u>out(24), 1, 1, 0, 0, 1, 0, 1, 0, 0, Parity) because it violates a Hamming weight</u>

<u>constraint of 9. The Algorithm of Figures 2-8 explicitly checks for Hamming</u>

weight violations even if there are no (0, 11/11) constraint violations to produce

an output with a Hamming weight of al least 9.

A.4.b (page 16 of the Appellant's Appeal Brief).

The Applicant contends, "These claims recite that the feature of processing the initial binary data comprises bitwise inverting the initial binary data if the Hamming weight of the initial binary data is less than the predetermined value. The Appellants submit that Nazari fails to teach or suggest such features".

The Examiner asserts that when dealing with Binary data replacing a 0 with a 1 or a 1 with a 0 is equivalent to inverting data. For example: if, in Nazari, IN = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), OUT = (1, 1, 1, 1, 1, 1, 1, 1, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 1, 1, 1, 1, 1, 1, 1, 1, Parity). The zeros in the first nine positions are replaced with ones which is substantially the same as inverting the first nine positions on a bit-by-bit basis since inverting the values in the first nine positions would have the same result, converting the first nine zeros to ones.

A.4.c (page 16 of the Appellant's Appeal Brief). The Applicant contends, "These claims recite that the feature of processing the initial binary data further comprises supplying an indication of whether the Hamming weight of the initial binary data is less than the predetermined value. The Appellants submit that Nazari fails to teach or suggest such features".

Note in particular; Nazari explicitly teaches that if the coding constraints of (0,11/11) and a Hamming weight of at least nine is satisfied only case 16 below is executed and the rest of the cases are ignored. Col. 2, lines 22-37 in Nazari explicitly teaches that the Tables A and B in Figure 2 are used to analyze the coding constraints to *determine* if there is a violation in the coding constraints. For Example; if IN = 00000010 00000001 00000010 00000001 then the odd and even interleaves are 0001 0000 0001 0000 and 0000 0001 0000 0001. Neither IN nor its odd and even interleaves violate the (0,11/11) constraint since there is not a run of 11 consecutive zeros in either IN or its odd and even interleaves. The algorithm still flags IN = 00000010 00000001 00000010 00000001 for further processing since it violates a constraint requirement of at least 9 ones. IN is converted to OUT= (0, 1, 1, 0, 1, 0, 0, 1, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 1, 0, 0, 1, 0, 1, 0, 0, Parity) because it violates a Hamming weight constraint of 9. The Algorithm of Figures 2-8 explicitly checks for Hamming weight violations even if there are no (0, 11/11) constraint violations to produce an output with a Hamming weight of al least 9.

Results of the analysis are a clear indication of a Hamming constrain violation, that is, if the number of ones falls below a predetermined value.

A.4.d (page 17 of the Appellant's Appeal Brief). The Applicant contends, "These claims recite the feature that the indication comprises a binary digit having a first value if the Hamming weight of the initial binary data is less than the predetermined value and

having a second value otherwise. The Appellants submit that Nazari fails to teach or

suggest such features".

OUT(17) is a binary value used to flag violations in the constraints. See Steps 13 and

14 in Figure 4 of Nazari.

A.4.e (page 17 of the Appellant's Appeal Brief). The Applicant contends, "These claims

recite the feature of supplying an indication of, whether bits of the processed binary data

are inverted. The Appellants submit that Nazari fails to teach or suggest such features".

The Examiner asserts that when dealing with Binary data replacing a 0 with a 1 or a 1

with a 0 is equivalent to inverting data. For example: if, in Nazari, IN = (0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), OUT = (1, 1, 1, 1, 1,

1, 1, 1, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 1, 1, 1, 1, 1, 1, 1, 1, Parity).

The zeros in the first nine positions are replaced with ones which is substantially the

same as inverting the first nine positions on a bit-by-bit basis since inverting the values

in the first nine positions would have the same result, converting the first nine zeros to

ones.

The Flags out(22), out(20), out(18), out(17), out(16), out(14) and out(12) are indicators

in the output used to indicate which bits have been converted to ensure that the process

is reversible.

A.4.f (page 17 of the Appellant's Appeal Brief). The Applicant contends, "These claims

recite the feature that the indication comprises a binary digit having a first value if the

bits of the processed binary data are inverted and having a second value otherwise-

The Appellants submit that Nazari fails to teach or suggest such features".

The Examiner asserts that when dealing with Binary data replacing a 0 with a 1 or a 1

with a 0 is equivalent to inverting data. For example: if, in Nazari, IN = (0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), OUT = (1, 1, 1, 1, 1,

1, 1, 1, 1, out(10), 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, out(24), 1, 1, 1, 1, 1, 1, 1, 1, 1, Parity).

The zeros in the first nine positions are replaced with ones which is substantially the

same as inverting the first nine positions on a bit-by-bit basis since inverting the values

in the first nine positions would have the same result, converting the first nine zeros to

ones.

OUT(17) is a binary value used to flag violations in the constraints. If any of the input

bits are converted it will have a vaue of zero.

A.4.g (page 7 of the Appellant's Appeal Brief). The Applicant contends, "These claims

recite the feature that the processing the initial binary data comprises performing at

least one of error correction coding, run-length encoding, and precoding. The Appellants

submit that Nazari fails to teach or suggest such features".

The Abstract in Nazari teaches Run-Length-Limited (RLL) encoding.

A.4.h (page 18 of the Appellant's Appeal Brief). The Applicant contends, "These claims

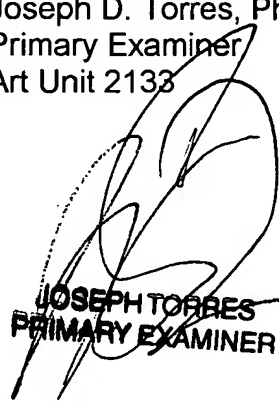recite the feature that a symbol boundary of an encoded symbol does not change

relative to error correction coding. The Appellants submit that Nazari fails to teach or

suggest such features".

A binary digit is a symbol an the binary digit is both its own end and start point, which

never changes regardless of any error correction encoding.

For the above reasons, it is believed that the rejections should be sustained.

                                                    Respectfully submitted,

                                                    Joseph D. Torres, PhD
                                                    Primary Examiner
                                                    Art Unit 2133

JT
August 22, 2005

Conferees
Albert Decady
Supervisory Primary Examiner
Art Unit 2133   JOSEPH TORRES
                                                    PRIMARY EXAMINER

Christine Tu
Primary Examiner   CHRISTINE T. TU
Art Unit 2133   Primary Examiner
                                                    ALBERT DECADY
                                                    SUPERVISORY PATENT EXAMINER
                                                    TECHNOLOGY CENTER 2100

KATTEN MUCHIN ROSENMAN LLP (MARVELL)
IP DOCKET
1025 THOMAS JEFFERSON STREET, N.W.
SUITE 700, EAST LOBBY
WASHINGTON, DC 20007-5201